公安院校
招录培养体制改革
试点专业
系列教材

计算机犯罪侦查方向

丛书主编 李锦

Oracle数据库应用与安全管理

闫薇 编著

清华大学出版社

公安院校招录培养体制改革试点专业系列教材

Oracle 数据库应用与安全管理

闫　薇　编著

清华大学出版社
北　京

内 容 简 介

本书精选数据库原理和 Oracle 数据库的核心内容，将抽象的理论知识用丰富的图示和通俗易懂的语言描述出来，采用案例教学的方式撰写，合理地组织学习单元，在实例上侧重实用性和启发性。本书内容丰富、深入浅出、通俗易懂、注重实用。

全书以 Oracle 11g for Windows 7 为平台，详细地讲解了数据库系统原理、Oracle 数据库应用及安全管理等内容。数据库系统原理部分主要介绍数据库系统的基本概念和基本理论，具体包括数据库系统的发展历程、数据库系统的结构、数据模型、关系代数运算、关系数据库标准语言 SQL、关系数据库规范化理论、数据库设计等内容。Oracle 数据库应用部分主要介绍 Oracle 数据库的基本原理及其应用技术，具体包括 Oracle 数据库的体系结构、PL/SQL 概述、异常处理、游标、存储过程、存储函数、包、触发器等内容。安全管理部分主要介绍用户管理、权限管理、角色管理、Oracle 数据库的备份与恢复方法等内容。

本书为教师配备了电子教案，在各章节配有上机实验练习和课后习题，并给出了实验练习参考答案和课后习题参考答案，以方便教师教学和学生自学自测。本书既可作为高等学校计算机专业、网络专业和信息管理等相关专业的数据库原理与应用课程的教材，也可以作为从事相关专业的工程技术人员和科研人员的参考资料。

图书在版编目（CIP）数据

Oracle 数据库应用与安全管理/闫薇编著. --北京：清华大学出版社，2015
公安院校招录培养体制改革试点专业系列教材
ISBN 978-7-302-39454-9

Ⅰ. ①O…　Ⅱ. ①闫…　Ⅲ. ①关系数据库系统－高等学校－教材　Ⅳ. ①TP311.138

中国版本图书馆 CIP 数据核字(2015)第 036502 号

责任编辑：闫红梅　李晔
封面设计：
责任校对：时翠兰
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×230mm　　**印　　张**：22.5　　**字　　数**：490 千字
版　　次：2015 年 6 月第 1 版　　**印　　次**：2015 年 6 月第1 次印刷
印　　数：1～　000
定　　价：　.00 元

产品编号：062531-01



随着计算机技术与网络通信技术的发展，数据库技术已成为信息社会中对大量数据进行组织与管理的重要技术手段，是网络信息化管理系统的基础。在众多数据库系统中，Oracle 数据库是世界范围内性能最优异的数据库系统之一，广泛应用于各行各业，如政府、交通、公安、电信、金融、能源等部门，并已逐渐成为企业信息化建设的重要数据库平台，始终处于数据库领域的领先地位。

本书以 Oracle 11g for Windows 7 为平台，详细地讲解了数据库系统原理、Oracle 数据库应用及安全管理等内容。本书采用案例教学的方式撰写，合理地组织学习单元，在实例上侧重实用性和启发性。全书分为 14 章、28 个实验和 4 个附录。第 1 章主要介绍数据库的基本概念、数据管理技术的发展阶段、数据库系统的结构和数据模型。第 2 章主要介绍关系数据结构、关系数据操作、关系的完整性、传统的集合运算和专门的关系运算。第 3 章主要介绍 SQL 语言概述、数据定义、数据查询、数据操纵和视图。第 4 章主要介绍关系数据库规范化理论、数据库设计概述、系统规划阶段、需求分析阶段、概念结构设计、逻辑结构设计、物理结构设计、数据库的实施、数据库的运行和维护。第 5 章主要介绍 Oracle 数据库的物理存储结构、逻辑存储结构、内存结构、进程结构和数据库例程。第 6 章主要介绍 PL/SQL 简介、PL/SQL 变量、PL/SQL 运算符和函数、PL/SQL 条件结构和循环结构。第 7 章主要介绍异常的概述、预定义异常、非预定义异常和用户自定义异常的处理方法和步骤。第 8 章主要介绍游标的定义、显式游标和隐式游标的应用。第 9 章主要介绍存储过程的创建、存储过程的调用和存储过程的管理。第 10 章主要介绍存储函数的创建、存储函数的调用、存储函数的管理、存储过程与存储函数的区别。第 11 章主要介绍包的简介、包的创建与调用、包的重载、包的管理和 Oracle 内置包。第 12 章主要介绍触发器概述、语句级触发器、行级触发器、INSTEAD OF 触发器、系统事件与用户事件触发器、触发器的管理。第 13 章主要介绍数据库安全性概述、Oracle 的安全机制、数据库完整性控制。第 14 章主要介绍事务的定义、事务的特性和事务控制语句、数据库的恢复技术、Oracle 数据库的备份和恢复方法。

本书中的所有案例均来自附录 A 样本数据库中的学生-课程数据库、员工-部门数据库。附录 B 给出了 Oracle 11g 数据库的安装和卸载过程。附录 C 给出了上机实验练习的参考答案。附录 D 给出了课后习题的参考答案。

本书内容丰富，辽宁警察学院的杨虹、曾刚、张爽、唐红杰和冯晶莹老师分别编写了第

7、8、9、10、11 章。其余章节由辽宁警察学院的闫薇老师编写并对本书进行了统稿。由于时间和水平有限,难免有不妥之处,恳请读者批评指正。

编　者

2015 年 3 月

目录



第 1 章

数据库系统绪论

数据库技术产生于20世纪60年代末70年代初,其主要目的是有效地管理和存取大量的数据资源。数据库技术至今已走过了50多年的历程,特别是近20年,数据库技术及其应用得到了迅猛的发展。数据库系统从早期的层次数据库和网状数据库,发展到目前占主流地位的关系数据库,已形成了较为完整的理论体系。

随着计算机技术与网络通信技术的发展,数据库技术已成为信息社会中对大量数据进行组织与管理的重要技术手段,是网络信息化管理系统的基础。数据库系统已经成为现代计算机系统的重要组成部分。

1.1 数据库的基本概念

1.1.1 信息、数据和数据处理

信息是对现实世界中存在的客观实体、现象和关系进行描述的具有特定意义的数据,是经过加工处理的数据。

信息和数据是两个关系紧密的概念。从广义上讲,数据实际上就是描述客观事物的符号记录,例如,记录(张三,女,1996,辽宁)就是数据。文字、图形、图像、声音等都是数据。从狭义上讲,能够进入计算机并且能由计算机进行处理的信息就是数据。尽管数据与信息在概念上不尽相同,但在使用上人们并不需要严格去区分它们。

所谓数据处理,就是从已有数据出发,经过适当加工处理得到新的所需要的数据。数据加工处理一般分为数据计算和数据管理两部分。数据计算相对简单,而数据管理却比较复杂。在实践应用中,人们逐步认识到对数据的有效处理离不开对数据进行结构化的管理,数据管理是数据处理过程的主要内容与核心部分,数据处理在本质上可以看作是数据管理。

数据管理主要是指数据收集、整理、组织、存储、维护、检索和传送等相应操作,这些操作都是数据处理业务中重要和必不可少的基本环节。

1.1.2 数据库

"数据库"这一术语有很多种解释。从字面上来看,就是存放数据的仓库。从本质上讲,数据库(DataBase,DB)是一个长期存储在计算机内、有组织的和可共享的大量数据集合。

数据库本身可以看作是一个具有高度数据集成性质的电子文件柜，它是基于计算机系统的持久性数据的“仓库”或者“容器”。

1.1.3 数据库管理系统

数据库管理系统(DataBase Management System，DBMS)是位于用户应用程序与操作系统之间的一层数据管理软件。DBMS 是数据库管理的中枢机构，是数据库系统具有数据共享、并发访问和数据独立性的根本保证。对数据库的所有管理包括定义、查询、更新和各种运行都需要通过 DBMS 实现。DBMS 通过提供相应的数据子语言(Data Sublanguage)来实现上述重要功能。

1. DBMS 中的数据子语言

DBMS 提供的数据子语言可以分为三类。

(1) 数据定义语言(Data Definition Language，DDL)：负责数据的模式定义与数据的物理存取构建。

(2) 数据操作语言(Data Manipulation Language，DML)：负责数据的操纵处理，例如查询、增加、删除和修改等。

(3) 数据控制语言(Data Control Language，DCL)：负责数据完整性和安全性的定义与检查，同时完成并发控制和恢复等职能。

以上语言都是非过程性语言，它们具有两种表现形式。

(1) 交互型命令语言：这种方式语言结构简单，可以在终端上实时操作，又称为自主型语言。

(2) 宿主型语言：应用这种方式，一般是将其嵌入在某些宿主语言(Host Language)当中，如 Fortran、C、C++ 等高级过程性语言中。

2. DBMS 的基本功能

DBMS 主要实现对数据的有效组织、管理和存取，因此 DBMS 具有以下基本功能：

(1) 数据定义功能。DBMS 提供相应数据语言来定义数据库结构，它们用于刻画数据库框架，并被保存在数据字典中。

(2) 数据存取功能。DBMS 提供数据操纵语言(DML)，实现对数据库数据的基本存取操作：检索、插入、修改和删除。

(3) 数据库运行管理功能。DBMS 提供数据控制功能，即是通过保证数据的安全性、完整性和并发控制等，实现对数据库的有效控制和管理，以确保数据正确有效。

(4) 数据库的建立和维护功能。包括数据库初始数据的装入，数据库的转储、恢复、重组织，系统性能监视、分析等功能。

(5) 数据库的传输。DBMS 提供处理数据的传输，实现用户程序与 DBMS 之间的通信，通常与操作系统协调完成。

1.1.4　数据库系统

数据库系统(DataBase System,DBS)是指引入数据库技术后的整个计算机系统,能够实现有组织地、动态地存储大量相关数据,提供数据处理和信息资源共享的核心系统。

数据库系统是具有数据库管理功能的计算机系统。作为一个系统,DBS 实际上是一个在计算机上可运行的、为应用系统提供数据并进行数据存储、维护和管理的系统,是存储介质、处理对象和管理系统的集合体。这里所说的“集合体”主要包括 3 个部分:计算机系统(软件、硬件和人)、数据库、数据库管理系统,即 DBS＝计算机系统＋DB＋DBMS,如图 1.1 所示。

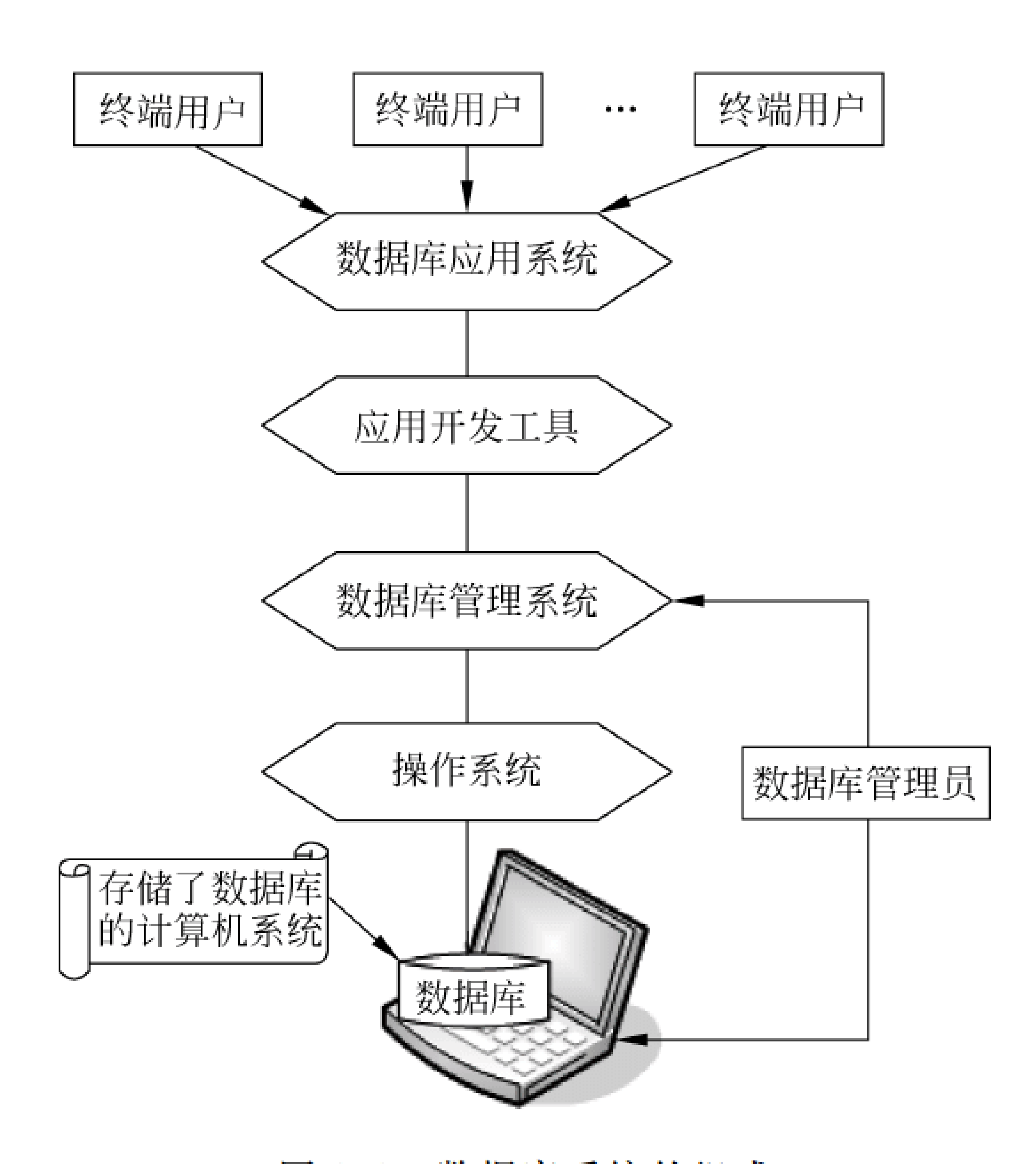


图 1.1　数据库系统的组成

1. 计算机系统

1) 软件支持平台

软件支持平台主要包括操作系统、各种主语言和应用开发支撑软件等。首先,DBMS 只有在操作系统支持下才能工作;其次,为了开发应用系统,需要各种主语言,如程序设计语言(C、Java、C++)以及与 Internet 有关的 HTML 和 XML 等;再就是为应用开发人员提供的高效、多功能的交互式程序设计系统,如可视化开发工具.NET 等。

2) 硬件支持系统

硬件支持系统是指数据存储和数据处理所必不可少的硬件设施,主要有如下几种:

(1) 中央处理器和相应的主存储设备——主要用于支持数据库系统软件的执行。

(2) 二级存储设备——其中包括相关的 I/O 设备(磁盘驱动器等)、设备控制器和 I/O 通道、必要的后备存储设备等。二级存储设备(大部分为磁盘)用来存放数据。

(3) 网络——过去数据库一般建在单机之上,现在较多建在网络之中,从发展趋势来看,数据库系统今后以建在网络中为主,其结构形式又以客户/服务器(C/S)方式和浏览器/服务器(B/S)方式为主。

3) 用户系统

DBS 中的人系统也称为数据库用户系统。数据库用户系统由三类人员组成。

(1) 应用程序员:负责编写数据库应用程序,这些程序通过向 DBMS 发出数据库操作语句请求访问数据库,这些程序通常是具有批处理特征或者联机特征的应用程序,目的是允许最终用户通过联机工作站或者终端访问数据库。

(2) 最终用户:通过联机工作站或者终端与数据库系统进行交互。实行交互的应用软件为数据库系统本身固有的,无须用户自己编写。

(3) 数据库管理员:由于数据库的共享性,数据库的规划、设计、维护和监视须由专人管理,这就是数据库管理员(DataBase Administer,DBA)。从数据库技术运行的角度来说,数据库管理员是三类用户中的灵魂人物。DBA 需要根据企业的数据情况与要求,制定数据库建设与维护的策略,并对这些策略的执行提供技术支持。数据库管理员负责技术层的全局控制。

2. 数据库

相互关联的且具有最小冗余的数据在数据库中按照一定物理组织结构存放,并且从用户和数据库管理系统角度来看,这些数据又是按一定逻辑结构组织的。这种物理组织结构和逻辑组织结构在最大程度上与用户所编制的应用程序相互独立。

3. 数据库管理系统

它一方面负责对数据库中的数据进行管理和维护;一方面为用户操作数据库中的数据提供一种公用的操作方法,接收用户的操作命令,帮助完成对数据库的操作并保障数据库的安全。目前,常见的数据库管理系统如 Oracle、SQL Server、MySQL 等。

综上所述,数据库、数据库管理系统和数据库系统是 3 个不同的概念。数据库强调的是数据,数据库管理系统强调的是系统软件,而数据库系统强调的是整个系统。数据库系统的目的在于维护信息,并在必要时提供协助来获取这些信息。另一方面,用户的目的是使用数据库,而数据库管理系统是帮助达到这一目的的工具和手段。

需要指出的是,人们常常将数据库作为数据库系统的同义词使用,将数据库系统作为数据库管理系统的同义词使用。

1.2 数据管理技术的发展阶段

从数据处理角度来看,基于计算机的数据管理技术经历了“人工管理”、“文件管理”和“数据库管理”三个阶段。

1.2.1　人工管理阶段

1. 人工管理数据的特点

数据的人工管理阶段出现在 1953—1955 年。在此期间,人们逐步认识到:数据管理中有许多工作是机械和重复的,而机械的事情当然最适合于使用机器来做,因此使用计算机管理数据就成为一种自然的考虑与趋势。当时硬件状况是没有可供直接访问的磁盘等存储设备,外存只有卡片机、磁带机;没有键盘、鼠标,只有"开始"、"停止"等简单控制计算机的按钮。在软件环境中,没有通用的操作系统,只有汇编语言;没有数据管理方面的软件,只有数据批处理方式。

2. 人工管理数据的优势与缺陷

人工管理的优点是使用计算机管理数据,速度加快,效率提高。

人工管理的特征是数据的外在物理结构与用户观点的逻辑结构完全一致,也就是说,数据的存储与数据的使用直接对应,基于物理方式存取数据,没有物理手段以外的其他访问方法和技术,用户必须掌握数据在计算机中确切的存储地址和存取方式。

人工管理数据是人们借助计算机进行数据管理的首次尝试,在计算机应用发展上具有重要的意义。当然,从现今的观点看,人工管理技术还存在不少缺陷,这主要表现为以下几方面:

(1) 数据不保存,没有持久性。由于数据主要用于科学计算,一般不需要将数据长期保存。计算某一问题时将数据输入,计算完毕之后就将数据删除,对于用户提供的数据是如此处理,对于系统软件运行过程中产生的数据也是同样处理。

(2) 数据依赖程序,缺乏独立性。由于没有相应软件系统完成数据的管理工作,所以应用程序不仅要规定好数据的逻辑结构,还要设计出数据物理结构,如存储结构、存取路径和输入方法等。

(3) 数据无共享,冗余度大。由于数据面向应用程序,一组数据只能对应一个程序。在出现多个不同程序涉及相同数据这一常见现象时,必须各自定义,否则难以相互参照利用,造成程序之间大量数据冗余。

(4) 程序管理数据,加重用户负担。由于应用程序管理数据,当数据的逻辑结构和物理结构变动时,必须对应用程序进行相应改变,使用户的负担相当沉重。

人工管理阶段应用程序与数据间的对应关系如图 1.2 所示。

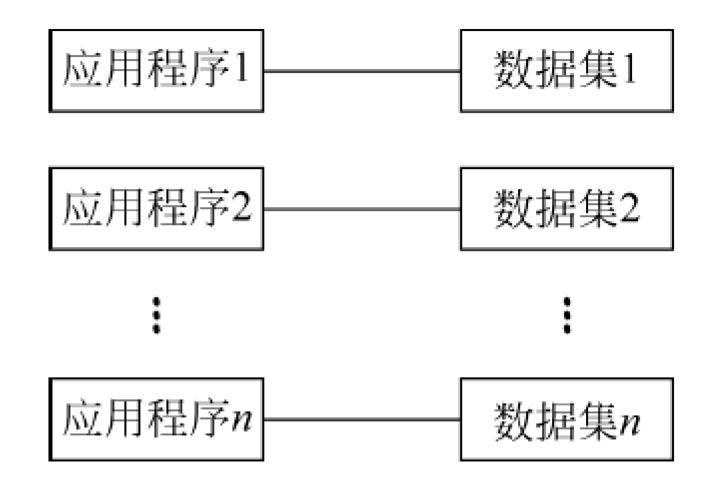


图 1.2　人工管理程序与数据的关系

1.2.2 文件系统管理阶段

采用文件系统处理数据是从20世纪50年代中期到60年代中期,大约10年左右。在这一时期,计算机不仅用于科学计算,同时也开始用于信息处理。由于信息量逐渐增加,数据存储、数据检索和数据维护已经成为实际应用中的迫切需要,随之而来的就是数据结构和数据管理技术的兴起与发展。在这个阶段,硬件有了很大的改进,出现了磁盘、磁鼓等直接存储设备。软件方面,高级语言和操作系统相继出现;而且在操作系统中也有了专门的数据管理软件,一般将其称为文件系统;数据处理不仅有了批处理的作业方式,还有了共享的实时处理方式。

1. 文件系统管理技术基本特征

文件系统管理技术的基本特征是改变了数据与用户的直接对应,文件的物理结构与文件的逻辑结构实现了初步分离,在文件物理结构中增加了链接和索引形式,可以对文件中的记录进行顺序和随机访问,并且提供各种应用程序对文件进行查询、修改、插入和删除等操作。文件管理技术为操作系统组成部分中的文件管理软件提供了从逻辑文件到物理文件的"访问手段",这种新的访问手段带来了数据管理的新特点。

(1) 数据长期保存:由于大量使用计算机进行数据处理,而其中一个关键问题就是需要反复进行查询、更新(插入、删除和修改)等基本操作,因此,数据就以文件形式被长期保存在计算机外部存储设备当中。

(2) 数据组织成相互独立的数据文件:改变了人工管理阶段"按(地)址存取"的方式,实现了"按(文件)名存取"。

(3) 应用程序与数据文件间存在多对多的关系:一个应用程序可以使用多个数据文件,一个数据文件也可以被多个应用程序所使用。

(4) 具有一定的共享性:此时文件共享只是对于某一类应用而言,范围不够广泛,距人们对数据处理所期望的共享性尚有距离。

2. 文件系统的缺陷

在文件系统管理阶段,由于设备具有独立性,当改变存储设备时,可以不改变应用程序,从而出现了数据管理的初级阶段。但此时还未真正实现在用户观点下的数据内在逻辑结构独立于数据外部物理结构的要求,数据物理结构变动时,用户的数据应用程序仍然需要改变,应用程序具有"程序—数据依赖性",有关物理表示的知识与访问技术还没有直接体现在应用程序的编码中。因此,文件系统管理技术存在着如下不足之处:

(1) 文件内有结构,整体无结构。文件内部记录之间具有必要的联系,但各个文件之间无联系,因此,局部有结构,整体无结构。

(2) 数据共享性差,存在较大冗余。由于文件之间没有结构,缺乏联系,所以每个应用程序都有自己对应的数据文件,同样数据有可能在多个文件中重复存储。

(3) 数据依赖应用,缺乏独立性。由于文件只能存储数据,不能存储文件记录的结构表

述，数据文件的基本操作都要依靠应用程序实现。

文件系统阶段中应用程序与数据之间的关系如图 1.3 所示。

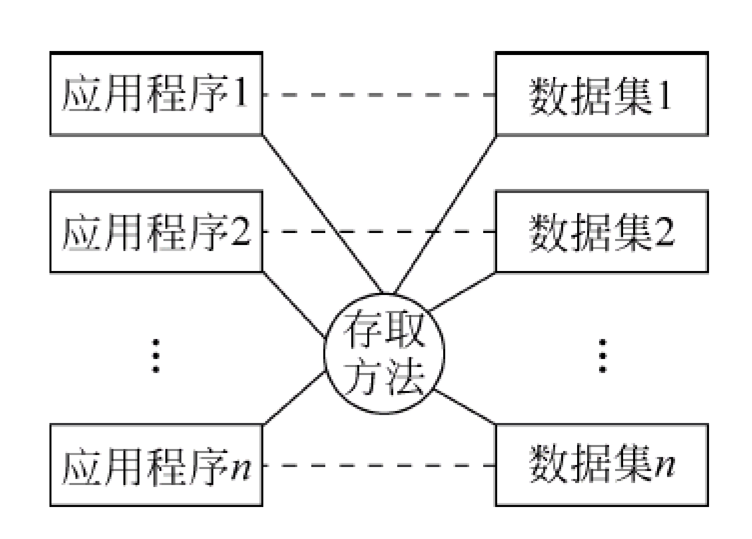


图 1.3　文件系统程序与数据的关系

1.2.3　数据库管理阶段

进入 20 世纪 60 年代，随着计算机应用领域的日益拓展，计算机用于数据管理的规模越来越大，基于文件系统的数据管理技术无法满足实际应用广泛而又迫切的需要。在这一时期，计算机硬件技术得到了飞速发展，大容量磁盘、磁盘阵列等基本的数据存储技术日益成熟，有效的存储硬件陆续进入市场，同时价格却在不断下降；同时许多厂家和公司竞相投入到数据管理技术的开发与研制当中，软件环境迅速完善。在迫切的实际需求和良好的硬件、软件环境中，数据库系统应运而生。

基于数据库系统的数据管理技术的本质是数据物理结构与数据逻辑结构的“完全”分离，通过数据库管理系统（DBMS）的统一监督和管理，使得所有应用程序中使用的数据汇集起来，按照一定结构进行组织和集成。与人工管理和文件系统管理阶段相比，数据库系统管理具有如下主要特点。

1. 数据高度结构化

数据结构化是数据库系统与文件管理系统的根本区别。数据库系统不仅要考虑数据项之间的联系，还要考虑数据类型之间的联系。例如，在一个企业中通常会有人事系统、工资系统、库存系统、销售系统等。如果采用文件系统进行管理，那么相应的数据都必须存放在相互分离的文件中。显然上述各个系统之间是有客观联系的，而使用数据库管理系统就人为地将它们割裂开来，使得相互联系只能通过应用程序才能体现出来。

数据库系统实现了整体数据的结构化，这是数据库的主要特征。在数据库系统中，数据不再针对某一项应用，而是面向应用的整体。

2. 数据共享性高，冗余度降低

数据库中的数据是高度共享的，也就是说，同一个用户可以以不同的应用目的访问同一数据；不同用户可以同时访问同一数据，即所谓的“并发访问”。数据的共享程度直接关系到数据的冗余程度。数据库系统是从整体构架来描述数据的，数据不再面向某个特定的应用程序而是面向整个系统，由此可以大大减少数据冗余，节省存储空间。

3. 高度的数据独立性

用户只需关注数据库名称、数据文件名称和文件中的属性名称等逻辑概念，而不用过多考虑数据的实际物理存储，也就是不需关心实际数据究竟存储在磁盘的什么位置。这样一来，数据与应用程序之间就具有了独立性，数据的定义和描述就可以从应用程序中分离出来；而且由于具有数据独立性，就有可能开发出专门用于数据管理的系统软件，即数据库管理系统，通过这个系统具体处理数据的存取路径等技术细节，从而简化了应用程序编写，减少了应用程序的维护和修改开销。

4. 具有专门的管理系统

数据库是一个多级系统结构，需要一组软件提供相应的工具进行数据的管理和控制，以达到保证数据安全性和一致性的基本要求。这样一组软件就是数据库管理系统(DataBase Management System，DBMS)。DBMS 的功能随着系统的不同而有所差异，但一般都具有数据的并发控制、数据的安全性保护、数据的完整性检查和数据库恢复等功能。

数据库系统阶段应用程序与数据间对应关系如图 1.4 所示。

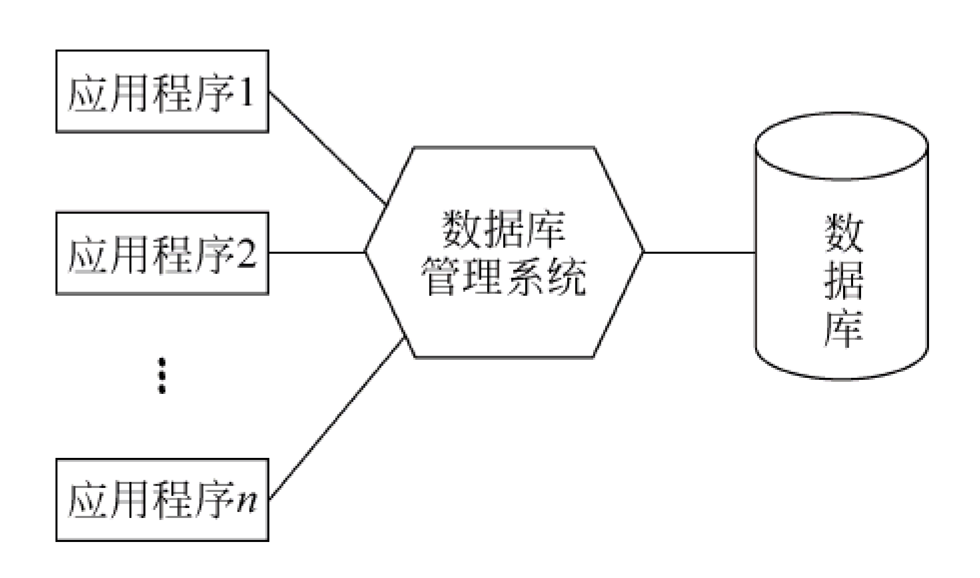


图 1.4　数据库系统程序与数据的关系

1.3　数据库系统的结构

1.3.1　模式结构

在数据模型中有“型”和“值”的概念。型是指对某一数据的结构和属性的说明，值是型的一个具体值。例如，学生(学号，姓名，性别，年龄，系别)是型，而(11432001，陈一，男，17，侦查系)是值。

模式是数据库中全体数据的逻辑结构和特征的描述，它只涉及型的描述，不涉及具体的值。模式是相对稳定的，而值是不断更新的。从数据库管理系统的角度看绝大多数数据库系统都采用三级模式结构，并提供两级映像功能。

1. 三级模式结构

数据库系统的三级模式是由外模式、模式和内模式构成，其结构如图 1.5 所示。

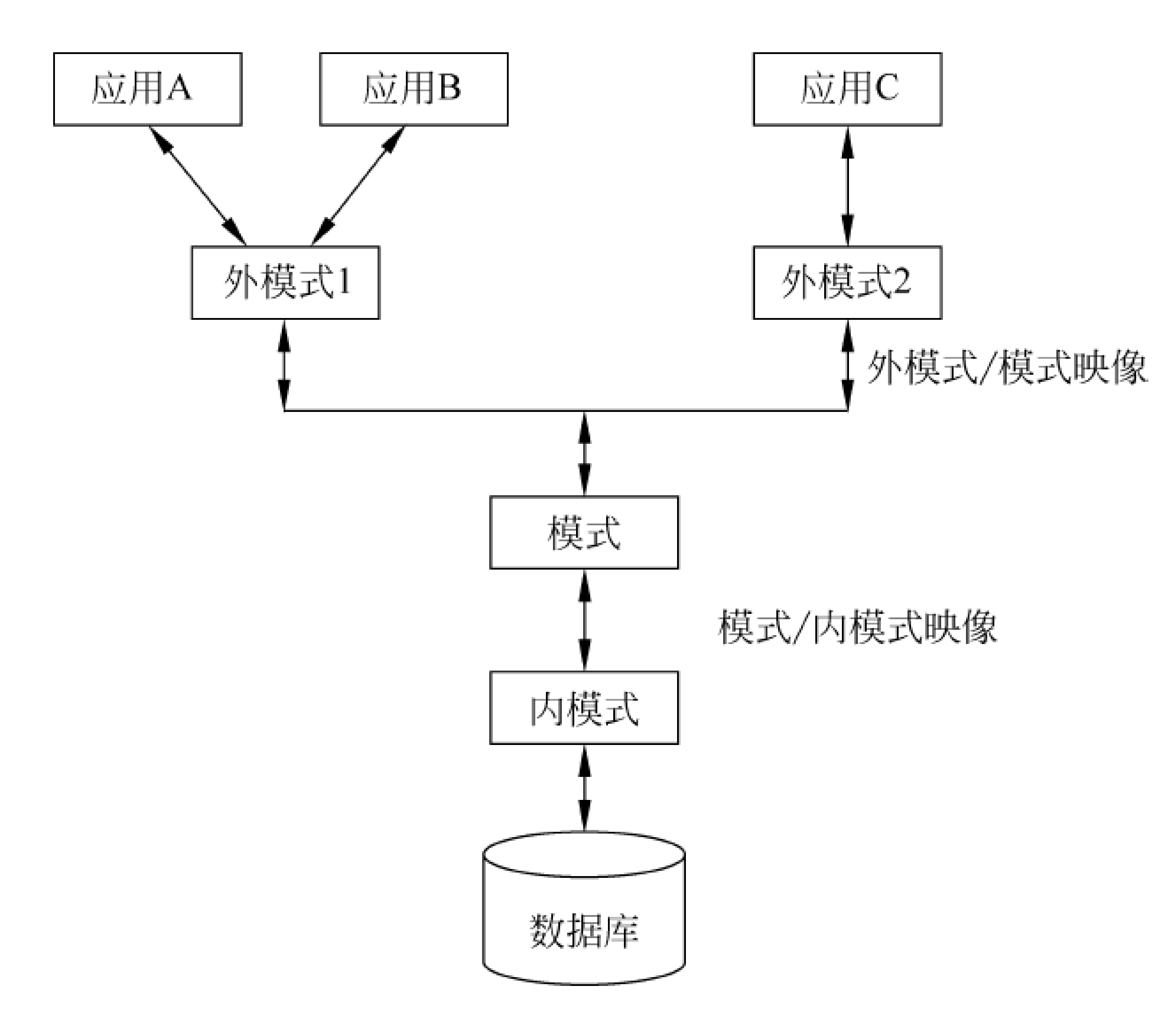


图 1.5　数据库系统的三级模式结构

1）内模式(Internal Schema)，也称存储模式

（1）内模式是数据在数据库系统的内部表示，即对数据的物理结构/存储方式的描述，是低级描述，一般由 DBMS 提供的语言或工具完成。

（2）通常我们不关心内模式的具体技术实现，而是从一般组织的观点(即概念模式)或用户的观点(外模式)来讨论数据库的描述。但我们必须意识到基本的内模式和存储数据库的存在。

（3）一个数据库只有一个内模式。

2）模式(Schema)，也称逻辑模式

（1）模式是数据库中全体数据的逻辑结构和特性的描述，是所有用户的公共数据视图。

（2）DBMS 提供数据定义语言 DDL 来描述逻辑模式，严格定义数据的名称、特征、相互关系、约束等。

（3）一个数据库只有一个模式。

3）外模式(External Schema)，也称用户模式

（1）外模式是模式的子集或变形，是与某一应用有关的数据的逻辑表示。

（2）不同用户的需求不同，看待数据的方式也可以不同，对数据保密的要求也可以不同，使用的程序设计语言也可以不同，因此不同用户的外模式的描述也可以是不同的。

（3）一个数据库有多个外模式。

2．两级映像功能

数据库系统的三级模式是对数据的三个抽象级别，为了能够实现这三个抽象层次的联系和转换，数据库管理系统在这三级模式之间提供了两层映像：外模式/模式映像、模式/内

模式映像。这两层映像保证了数据库系统的逻辑独立性和物理独立性。

1）外模式/模式映像

模式描述的是数据的全局逻辑结构，外模式描述的是数据的局部逻辑结构。对应于同一个模式可以有任意多个外模式。对于每一个外模式，数据库系统都有一个外模式/模式映像，它定义了该外模式与模式之间的对应关系。

2）模式/内模式映像

内模式描述的是数据的物理结构和存储方式，数据库中只有一个模式，也只有一个内模式，所以模式/内模式映像是唯一的，它定义了数据全局逻辑结构与存储结构之间的对应关系。

3. 数据独立性

数据独立性指应用程序和数据之间相互独立，不受影响。数据独立性分为逻辑独立性和物理独立性两级。

1）逻辑独立性

逻辑独立性是通过外模式/模式映像来实现的。外模式/模式映像定义了数据的全局逻辑结构和局部逻辑结构之间的对应关系。

当数据库的模式改变时（例如，增加新的数据类型、新的数据项、新的关系等），只需要数据库管理员对各个外模式/模式映像做相应改变，就可以使外模式保持不变。因为应用程序都是根据外模式编写的，外模式保持不变，则应用程序保持不变，从而保证了数据和应用程序的逻辑独立性。

2）物理独立性

物理独立性是通过模式/内模式映像来实现的。模式/内模式映像定义了数据的全局逻辑结构和数据的存储结构之间的对应关系。

当数据库的存储结构改变时（例如，采用了更先进的存储结构），只需要数据库管理员对模式/内模式映像做相应改变，就可以使模式保持不变。由于外模式是模式的子集，模式保持不变，外模式也保持不变，则根据外模式编写的应用程序也保持不变，因此应用程序可以不必修改，从而保证了数据和应用程序的物理独立性。

数据的独立性是数据库系统最基本的特征之一，采用数据库技术使得应用程序的维护工作量大大减轻，保证了应用程序的稳定性。

1.3.2 体系结构

前面介绍的数据库系统的三级模式结构是从DBMS的角度看的数据库系统结构。若从用户的角度看，数据库系统的体系结构又可分为单用户数据库系统、多用户数据库系统、分布式结构的数据库系统、客户/服务器结构的数据库系统、浏览器/服务器结构的数据库系统等。

1. 单用户数据库系统

单用户数据库系统是最简单的数据库系统。整个数据库系统,包括应用系统、数据库管理系统、数据库等都装在一台计算机上,由一个用户独占,不同机器之间无法实现数据共享。采用该体系结构的应用系统通常会造成大量数据的冗余。

2. 多用户数据库系统

多用户数据库系统是指一个主机带多个终端的多用户结构的数据库系统。在这种结构下,应用程序、数据库管理系统、数据库等都集中存放在主机上,所有处理任务都由主机来完成,各个用户通过主机的终端并发地存取数据库中的数据,达到共享数据资源的目的。

多用户数据库系统的优点是数据易于管理与维护。缺点是对用户数量有限制,因为当主机的任务过于繁重时,主机可能成为瓶颈,从而使系统性能大幅度下降。另外,由于集中管理,当主机出现故障时,整个系统将瘫痪,因此对系统的可靠性要求较高。

3. 分布式结构的数据库系统

分布式结构的数据库系统是指数据库中的数据在逻辑上是一个整体,但物理地分布在计算机网络的不同结点上。网络中的每个结点都可以独立处理本地数据库中的数据,执行局部应用;同时也可以同时存取和处理多个异地数据库中的数据,执行全局应用。

分布式结构的数据库系统的优点是适应了地理上分散的公司、团体和组织对于数据库应用的需求。缺点是数据的分布式存放给数据的处理、管理与维护带来困难。当用户需要经常访问远程数据时,系统效率会明显地受到网络传输的制约。

4. 客户/服务器结构的数据库系统

在客户/服务器(Client/Server,C/S)结构的数据库系统中,数据处理任务被划分为两部分:一部分运行在客户端,另一部分运行在服务器端。划分的方案可以有多种,一种常用的方案是:客户端负责应用处理,数据库服务器完成 DBMS 的核心功能。

在 C/S 结构中,客户端软件和服务器端软件可以运行在一台计算机上,但大多是分别运行在网络中不同的计算机上。客户端软件一般运行在 PC 上,服务器端软件可以运行在从 PC 到大型机等各类计算机上。数据库服务器把数据处理任务分开在客户端和服务器上运行,因而充分利用了服务器的高性能数据库处理能力以及客户端灵活的数据表示能力。通常从客户端发往数据库服务器的只是查询请求,从数据库服务器传回给客户端的只是查询结果,不需要传送整个文件,从而大大减少了网络上的数据传输量。

客户/服务器结构的数据库系统的优点是显著减少了网络上的数据传输量,提高了系统的性能、吞吐量和负载能力。另外,客户/服务器结构的数据库往往支持多种不同的硬件和软件平台,应用程序具有更强的可移植性,可以减少软件维护开销。

5. 浏览器/服务器结构的数据库系统

上述 C/S 结构是一个简单的两层模型,一端是客户机,另一端是服务器。这种模型中,客户机上都必须安装应用程序和工具,使客户端过于庞大、负担太重,而且系统安装、维护、升级和发布困难,从而影响效率。

随着 Internet 的迅速普及，出现了三层客户机/服务器模型：客户机→应用服务器→数据库服务器。这种结构的客户端只需安装浏览器就可以访问应用程序，这种系统称为浏览器/服务器(Browser/Server，B/S)系统。B/S 结构克服了 C/S 结构的缺点，是 C/S 的继承和发展。

1.4 数据模型

1.4.1 数据模型的概念

模型(Model)是现实世界特征的模拟和抽象。例如，火车模型是对生活中火车的一种模拟和抽象，它可以模拟火车的启动、加速、减速和停车，它抽象了火车的基本特征——车头、车身、车尾。

数据模型(Data Model)也是一种模型，用来描述数据、组织数据和对数据进行操作。

由于计算机不可能直接处理现实世界中的具体事物，所以人们必须事先把具体事物转换成计算机能够处理的数据。也就是首先要数字化，把现实世界中具体的人、物、活动、概念用数据模型这个工具来抽象、表示和处理。人们首先把现实世界抽象为信息世界，然后将信息世界转换为机器世界。这一过程如图 1.6 所示。

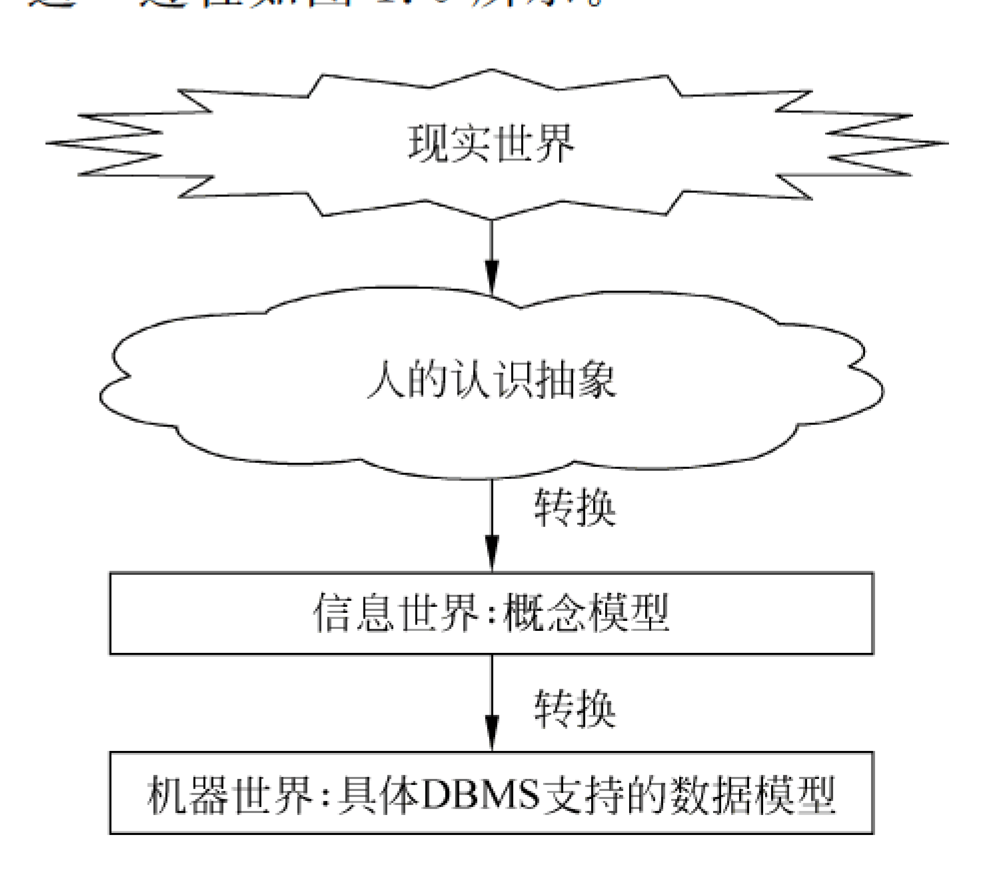


图 1.6 现实世界中客观对象的抽象过程

(1) 现实世界：客观存在的事物及联系。

(2) 概念世界：对现实世界的认识和抽象描述，按用户的观点对数据和信息建模，用于数据库设计。

(3) 机器世界：建立在计算机上的数据模型，按计算机系统的观点对数据建模，用于 DBMS 的实现。

数据模型是现实世界数据特征的抽象，它应满足三方面要求：

(1) 能比较真实地模拟现实世界。

(2) 容易为人所理解。

(3) 便于在计算机上实现。

一种数据模型要很好地满足上述三方面的要求在目前比较困难。在数据库系统中针对不同的使用对象和应用目的,采用不同的数据模型。因此,可以将数据模型划分为两类,它们分属于两个不同的层次:第一类是概念模型,第二类是逻辑模型和物理模型。

第一类概念模型,也称信息模型,它是按用户的观点来对数据和信息建模,主要用于数据库设计。

第二类中的逻辑模型主要包括层次模型(Hierarchical Model)、网状模型(Network Model)、关系模型(Relational Model)等,它是按计算机系统的观点对数据建模,主要用于DBMS 的实现。

第二类中的物理模型是对数据最低层的抽象,它描述数据在系统内部的表示方式和存取方法,在磁盘或磁带上的存储方式和存取方法,是面向计算机系统的。物理模型的具体实现是 DBMS 的任务,数据库设计人员要了解和选择物理模型,一般用户则不必考虑物理级的细节。

数据模型是数据库系统的核心和基础。各种机器上实现的 DBMS 软件都是基于某种数据模型的。

从现实世界到概念模型的转换是由数据库设计人员完成的,这个转换并不依赖于具体的计算机系统,而是概念级的模型;从概念模型到逻辑模型的转换可以由数据库设计人员完成,这种转换依赖于计算机上 DBMS 所支持的数据模型;从逻辑模型到物理模型的转换一般是由 DBMS 完成的。

1.4.2　数据模型的三要素

数据模型通常由数据结构、数据操作和数据的完整性约束三部分组成。

1. 数据结构

数据结构用于描述系统的静态特征,是描述数据库的组成对象以及对象之间的联系。数据结构是刻画一个数据模型性质最重要的方面。因此,在数据库系统中,人们通常按照其数据结构的类型来命名数据模型。例如层次结构、网状结构和关系结构的数据模型分别命名为层次模型、网状模型和关系模型。

2. 数据操作

数据操作用于描述系统的动态特性,是指对数据库中各种对象所允许执行的操作的集合,包括操作及有关的操作规则。数据库主要有检索和更新(包括插入、删除、修改)两大类操作。数据模型必须定义这些操作的确切含义、操作符号、操作规则(如优先级)以及实现操作的语言。

3. 数据的完整性约束

数据的约束条件是一组完整性规则的集合。完整性规则是给定的数据模型中数据及其联系所具有的制约和依存规则,用以限定符合数据模型的数据库状态以及状态的变化,以保证数据的正确、有效、相容。

数据模型应该反映和规定本数据模型必须遵守的基本的通用的完整性约束条件。例如,在关系模型中,任何关系必须满足实体完整性和参照完整性两个条件。

此外,数据模型还应该提供定义完整性约束条件的机制,以反映具体应用所涉及的数据必须遵守的特定的语义约束条件。例如,在学校的数据库中规定学生的考试成绩在0～100分之间,学生的入学年龄在16～35岁之间等。

1.4.3 概念模型

概念模型用于信息世界的建模,是现实世界到信息世界的第一层抽象,是数据库设计人员和用户之间进行交流的语言。

1. 基本概念

1) 实体(Entity)

客观存在并可相互区别的事物称为实体。实体可以是具体的人、事、物,也可以是抽象的概念或联系,例如,一个学生、一门课程、一次选课、一个部门、一个职工、公司与员工的雇用关系等都是实体。

2) 实体集(Entity Set)

同型实体的集合称为实体集。例如,全体学生就是一个实体集。

3) 实体型(Entity Type)

具有相同属性的实体必然具有共同的特征和性质。用实体名及其属性名集合来抽象和刻画同类实体,称为实体型。例如,学生(学号,姓名,性别,年龄,系别)就是一个实体型。

4) 属性(Attribute)

实体所具有的某一特性称为属性。一个实体可以由若干个属性来刻画。例如,学生实体可以由学号、姓名、性别、年龄、院系等属性组成,(11432001,陈一,男,17,侦查系)这些属性组合起来表示了一个学生。

5) 域(Domain)

属性的取值范围称为该属性的域。例如,姓名的域为字符串,年龄的域为小于35的整数,性别的域为(男,女)。

6) 码(Key)

唯一标识实体的属性集称为码。例如,学号是学生实体的码(学号可以唯一标识一个学生实体)。学号与课程号的组合是选修实体的码(学号与课程号的组合可以唯一标识一个学生与一门课程的一次选修关系)。

7）联系(Relationship)

在现实世界中，事物内部以及事物之间是有联系的，这些联系在信息世界中反映为实体内部的联系和实体之间的联系。实体内部的联系通常是指组成实体的各属性之间的联系。实体之间的联系通常是指不同实体集之间的联系。

两个实体集之间的联系可以分为三类：

（1）一对一联系(1∶1)。

如果对于实体集 A 中的每一个实体，实体集 B 中至多有一个(也可以没有)实体与之联系，反之亦然，则称实体集 A 与实体集 B 具有一对一联系，记为 1∶1。

例如，一个班级只有一个班长，而一个班长只属于一个班级，则班级与班长之间具有一对一联系。

（2）一对多联系(1∶n)。

如果对于实体集 A 中的每一个实体，实体集 B 中有 n 个实体($n \geqslant 0$)与之联系，反之，对于实体集 B 中的每一个实体，实体集 A 中至多只有一个实体与之联系，则称实体集 A 与实体集 B 有一对多联系，记为 1∶n。

例如，一座城市拥有多条街道，而每条街道只能属于一座城市，则城市与街道之间具有一对多联系。

（3）多对多联系(m∶n)。

如果对于实体集 A 中的每一个实体，实体集 B 中有 n 个实体($n \geqslant 0$)与之联系，反之，对于实体集 B 中的每一个实体，实体集 A 中也有 m 个实体($m \geqslant 0$)与之联系，则称实体集 A 与实体集 B 具有多对多联系，记为 m∶n。

例如，一个学生可以选修多门课程，而每门课程可以被多个学生所选修，则学生与课程之间具有多对多联系。

可以用图形来表示两个实体集之间的三类联系，如图 1.7 所示。

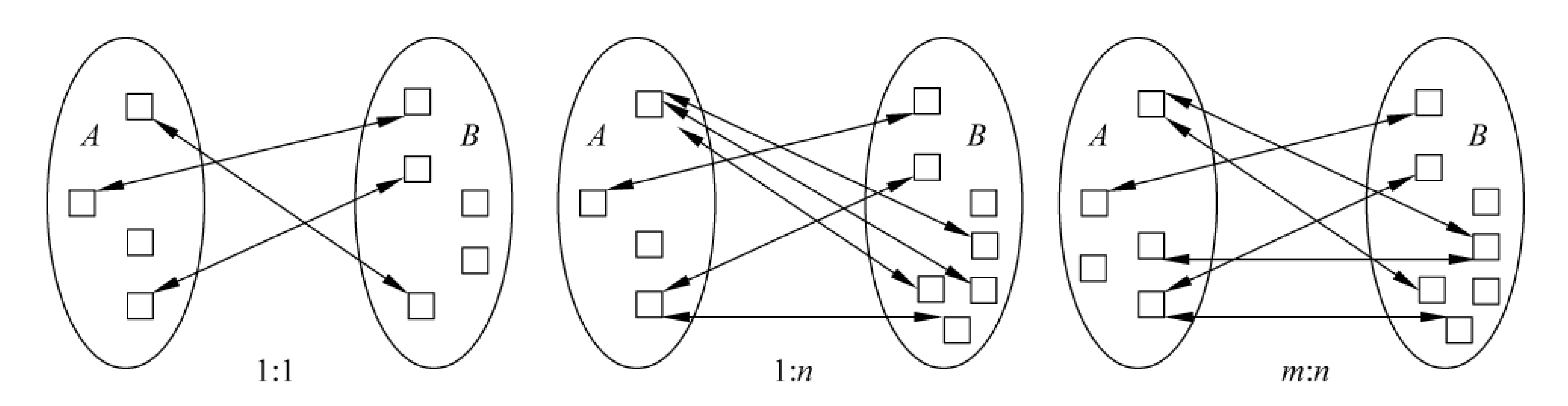


图 1.7　两个实体集之间的三类联系

2. 概念模型的 E-R 图表示

E-R 模型是由 P. P. S. Chen(美籍华人陈平山)于 1967 年提出的实体-联系方法(Entity-Relationship Approach)。由于它简单易学，因而在数据库系统应用的设计中，得到了广泛应用。该方法主要用 E-R 图(E-R Diagram)来描述现实世界的概念模型。E-R 图提供了表

示实体、属性和联系的方法。

（1）实体：用矩形表示，矩形框内写明实体名。

例如，学生实体和教师实体，用 E-R 图表示如图 1.8 所示。



图 1.8　学生实体及教师实体

（2）属性：用椭圆形表示，并用无向边将其与相应的实体连接起来。

例如，学生实体具有学号、姓名、年龄、性别、院系等属性，用 E-R 图表示如图 1.9 所示。

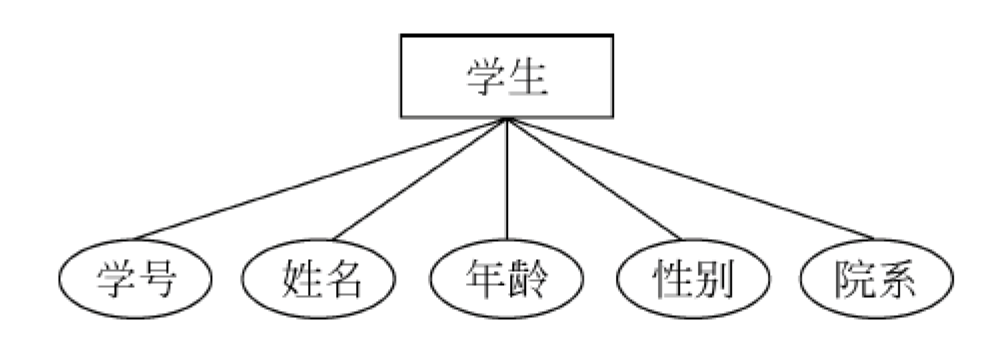


图 1.9　学生实体及属性

（3）联系：用菱形表示，菱形框内写明联系名，并用无向边分别与有关实体连接起来，同时在无向边旁标上联系的类型（1∶1，1∶n，m∶n）。三种基本联系，用 E-R 图表示如图 1.10 所示。

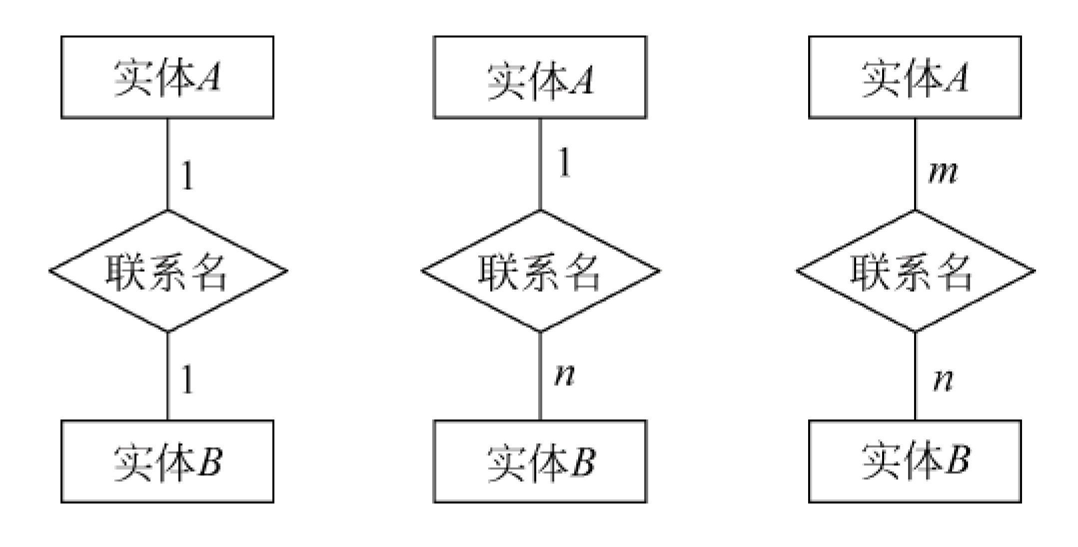


图 1.10　三种基本联系

例如，学生实体与课程实体之间有选修的联系，此联系为 m∶n 类型，用 E-R 图表示如图 1.11 所示。



图 1.11　学生实体与课程实体之间的联系

（4）直线：用无向边表示。需要注意的是，如果一个联系具有属性，则这些属性也要用无向边与该联系连接起来。

例如，如果用“成绩”来描述联系“选修”的属性，表示某个学生选修了某门课程的成绩。那么这两个实体及其之间的联系的 E-R 图表示如图 1.12 所示。

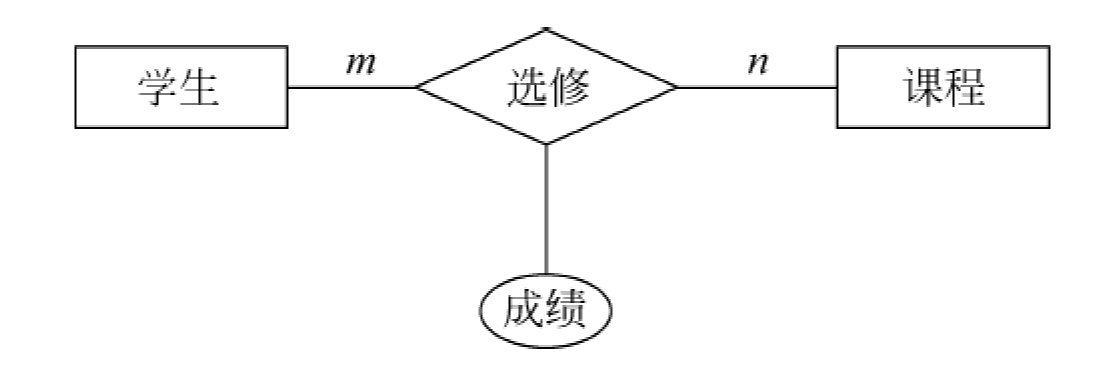


图 1.12　联系的属性

建立 E-R 图的步骤总结如下：

(1) 首先确定实体类型。

(2) 然后确定联系类型。

(3) 再把实体类型和联系类型组合成 E-R 图。

(4) 接着确定实体类型和联系类型的属性。

(5) 最后确定实体类型的码，并在属性名下画一条横线。

例题 1.1　假设学校管理规定：一个学生可以选修多门课程，一门课程可以有若干学生选修；一个教师可以讲授多门课程，一门课程只有一个教师讲授；一个学生选修一门课程，只有一个成绩。学生的属性有学号、学生姓名；教师的属性有教师编号、教师姓名；课程的属性有课程号、课程名。根据上述语义画出 E-R 图，要求在图中画出实体的属性、联系的类型以及实体的码。

下面给出学生选课系统的 E-R 图，如图 1.13 所示。

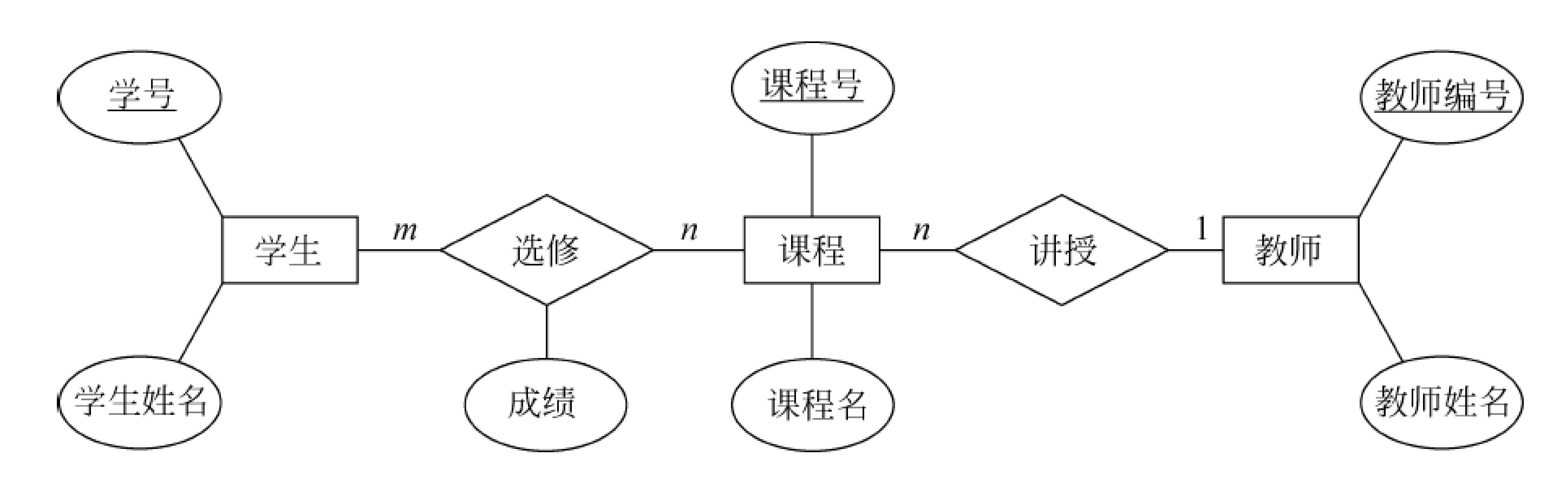


图 1.13　学生选课系统 E-R 图

例题 1.2　某个企业集团有若干工厂，每个工厂生产多种产品，且每个产品可以在多个工厂生产，每个工厂按照固定的计划数量生产产品；每个工厂聘用多名职工，且每个职工只能在一个工厂工作，工厂聘用职工有聘用期和工资。工厂的属性有工厂编号、厂名、地址，产品的属性有产品编号、产品名、规格，职工的属性有职工号、姓名。根据上述语义画出 E-R 图。在 E-R 图中需注明实体的属性、联系的类型以及实体的码。

下面给出工厂管理系统的 E-R 图，如图 1.14 所示。

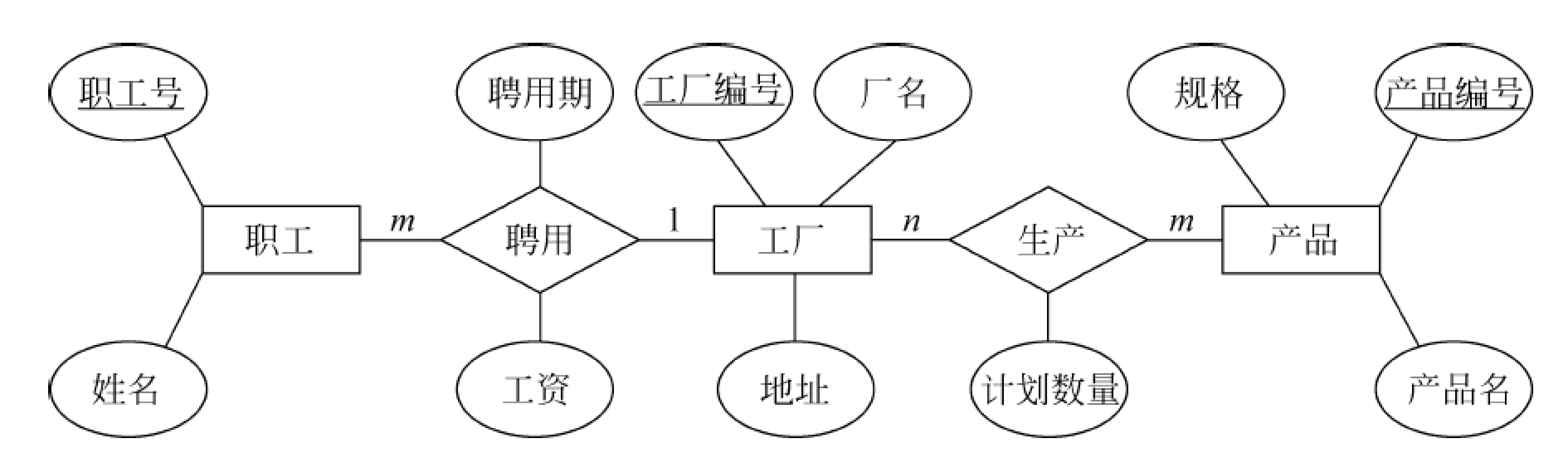


图 1.14 工厂管理系统 E-R 图

1.4.4 常用的数据模型

目前,数据库领域中常用的逻辑数据模型有层次模型(Hierarchical Model)、网状模型(Network Model)、关系模型(Relational Model)面向对象模型(Object Oriented Model)和对象关系模型(Object Relational Model),其中层次模型和网状模型统称为非关系模型。

1. 层次模型

层次模型用树形结构来表示各类实体以及实体间的联系。层次模型必须满足以下三个条件。

(1) 有且只有一个结点没有双亲结点,这个结点称为根结点。

(2) 根以外的其他结点有且只有一个双亲结点。

(3) 上一层结点和下一层结点间联系是 1∶n 联系。

图 1.15 是层次模型的一个例子,其中 $R1$ 为根结点;$R2$ 和 $R3$ 为兄弟结点,是 $R1$ 的子女结点;$R4$ 和 $R5$ 是兄弟结点,是 $R2$ 的子女结点;$R3$、$R4$ 和 $R5$ 为叶子结点。从图中可以看出层次模型是一棵倒立的树,结点的双亲是唯一的,所以层次模型也称为树状结构图。

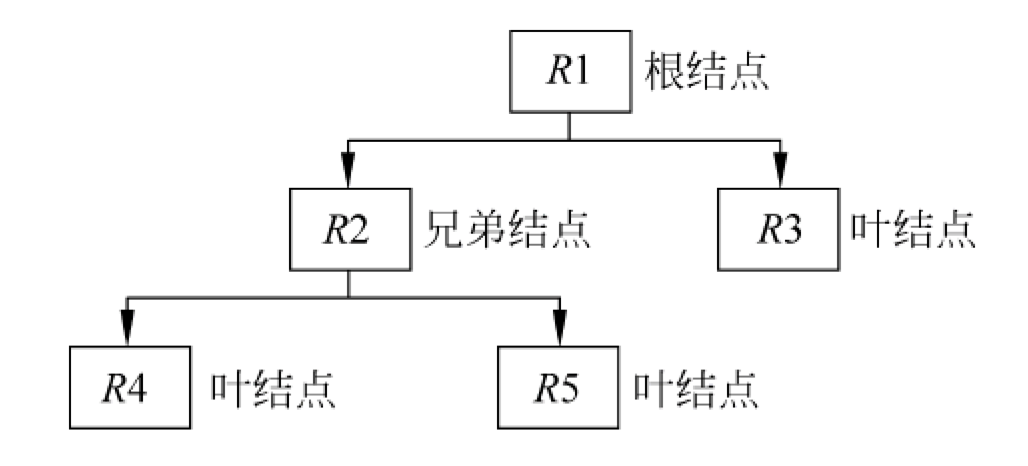


图 1.15 层次模型的例子

层次模型对具有一对多的层次关系的部门描述非常自然、直观,容易理解,如现实世界中的家庭和组织机构关系等适合用层次模型表示。

层次模型的优点是查询数据很简便,无须设计特别的算法,因为查询路径是唯一的,查询效率高;缺点是只能表示 1∶n 联系,无法描述事物之间复杂的联系。同时由于树状结构层次顺序严格和复杂,对数据插入和删除操作的限制比较多。

2. 网状模型

网状模型是用有向图表示实体及实体间联系的数据模型。网状模型必须满足以下三个条件。

(1) 有向图中的有向边表示从箭尾一端的结点到箭头一端的结点间的 1∶n 类型。将箭尾一端称为双亲结点,箭头一端称为子女结点。

(2) 允许一个以上的结点无双亲结点。

(3) 一个结点可以有多于一个的双亲结点,也可以有多于一个的子女结点。

网状模型的例子如图 1.16 所示,任一结点既可以有多个双亲结点,也可以有多个子女结点。

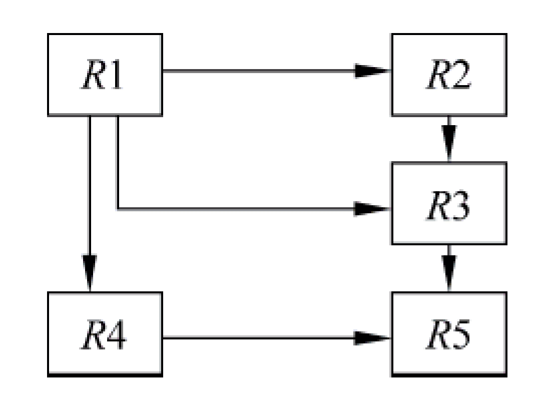


图 1.16　网状模型的例子

网状模型的优点是查询效率高,并能实现 m∶n 联系(每个 m∶n 联系可以分解成两个 1∶n 联系)。相比于层次模型,网状模型适合于描述较为复杂的现实世界;缺点是由于数据结构太复杂,不利于用户掌握。同时在有向图中可以到达某一个结点的路径有多条,因此应用开发者必须选择相对较优的路径来进行搜索以提高查询效率,这对应用开发人员的要求是较高的。

3. 关系模型

关系模型是目前最重要的一种数据模型。关系数据库系统采用关系模型作为数据的组织方式。

1970 年美国 IBM 公司 San Jose 研究室的研究员 E. F. Codd 首次提出了数据库系统的关系模型,开创了数据库关系方法和关系数据理论的研究,为数据库技术奠定了理论基础。由于 E. F. Codd 的杰出工作,他于 1981 年获得 ACM 图灵奖。

20 世纪 80 年代以来,计算机厂商新推出的数据库管理系统几乎都支持关系模型,非关系系统的产品也大都加上了关系接口。数据库领域当前的研究工作也都是以关系方法为基础。

1) 关系模型的数据结构

关系模型与以往的模型不同,关系模型主要是用二维表格结构表达实体及实体间联系的数据模型,在用户看来,关系模型中数据的逻辑结构(即数据结构)就是一张二维表,它由行和列组成,如表 1.1 所示。

表1.1 关系模型的数据结构(学生表)

学号	姓名	性别	年龄	系别
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
…	…	…	…	…

2) 关系模型的基本术语

(1) 关系(Relation):一个关系对应一张表,如学生表。

(2) 元组(Tuple):表中的一行即为一个元组。

(3) 属性(Attribute):表中的一列即为一个属性,给每一个属性起一个名称即属性名。如学生表对应五个属性:学号、姓名、年龄、性别和院系。

(4) 主码(Key):表中的某个属性组,它可以唯一确定一个元组,学生表中的学号,可以唯一确定一个学生,也就成为本关系的主码(或主键)。

(5) 域(Domain):属性的取值范围,如大学生年龄属性的域是(14～35),性别的域是(男,女),院系的域是一个学校所有院系的集合。

(6) 分量:元组中的一个属性值。

(7) 关系模式:对关系的描述,一般表示为:

关系名(属性1,属性2,…,属性n)

在关系模型中,实体以及实体间的联系都是用关系(即二维表)来表示。例如学生、课程、学生与课程之间的多对多联系在关系模型中可以表示如下:

学生(学号,姓名,年龄,性别,院系)
课程(课程号,课程名,学分)
选修(学号,课程号,成绩)

3) 关系数据模型的优缺点

关系数据模型的优点是关系模型与非关系模型不同,它是建立在严格的数学概念的基础上的;关系模型的概念单一,无论实体还是实体之间的联系都用关系(即二维表)表示,对数据的检索结果也是关系(即二维表),所以其数据结构简单、清晰,用户易懂易用;关系模型的存取路径对用户透明,从而具有更高的数据独立性、更好的安全保密性,也简化了程序员的工作和数据库开发建立的工作。

当然,关系数据模型也有缺点,其中最主要的缺点是,由于存取路径对用户透明,查询效率往往不如非关系数据模型。因此为了提高性能,必须对用户的查询请求进行优化,因此增加了开发数据库管理系统的难度。

正是由于上述的一些特点,关系数据库已经成为当代数据库技术的主流。而面向对象

模型和对象关系模型相对比较复杂，涉及的面较广，因此，目前面向对象和对象关系的数据库没有关系数据库那样普及。

1.5 本章小结

本章首先介绍了数据库的基本概念，包括信息、数据、数据处理、数据库、数据库管理系统和数据库系统的定义。简述了数据库、数据库管理系统和数据库系统之间的关系。通过对数据管理技术发展历程的回顾，阐述了发展到数据库系统阶段的必然性。

其次介绍了数据库系统的结构。从 DBMS 角度看，数据库系统具有三级模式和两级映像结构。从数据库最终用户的角度看，数据库系统的结构分为单用户结构、多用户结构、分布式结构、客户/服务器结构、浏览器/服务器结构等，这是数据库系统的外部体系结构。

最后介绍了数据模型，数据模型是数据库系统的核心和基础。数据模型是对现实世界进行抽象的工具，主要包括概念模型、逻辑模型和物理模型。组成数据模型的三个要素是数据结构、数据操作和完整性约束。

概念模型也称信息模型，用于信息世界的建模，E-R 模型是这类模型的典型代表，E-R 方法简单、清晰，应用广泛。

常用的逻辑数据模型有层次模型、网状模型、关系模型、面向对象模型、对象关系模型等。由于层次数据库和网状数据库已经逐渐被关系数据库代替，所以本书只在本章简单介绍了层次模型和网状模型的数据结构。目前，关系数据库已经成为当代数据库技术的主流，关系模型在这里也只是简单介绍，在后续章节中将详细介绍关系模型。

1.6 课后习题

一、选择题

1. 声音、文字、图形、病人的医疗记录、火车票的剩余情况等，这些都是(　　)。

A. 信息　　B. 数据　　C. 数据库　　D. 数据项

2. 数据库(DB)、数据库系统(DBS)和数据库管理系统(DBMS)之间的关系是(　　)。

A. DBMS 包括 DB 和 DBS　　B. DBS 就是 DB，也就是 DBMS

C. DB 包括 DBS 和 DBMS　　D. DBS 包括 DB 和 DBMS

3. 在数据管理技术发展所经历的三个阶段中，数据独立性最低的是(　　)。

A. 人工管理阶段　　B. 文件管理阶段

C. 计算机管理阶段　　D. 数据库管理阶段

4. 在数据管理技术发展所经历的三个阶段中，数据独立性最高的是(　　)。

A. 人工管理阶段　　B. 文件管理阶段

C. 计算机管理阶段　　D. 数据库管理阶段

5. 数据库系统的数据独立性体现在(　　)。

A. 不会因为数据的变化而影响到应用程序

B. 不会因为数据存储结构与数据逻辑结构的变化而影响应用程序

C. 不会因为存储策略的变化而影响存储结构

D. 不会因为某些存储结构的变化而影响其他的存储结构

6. 数据库系统的体系结构为三级模式两级映像,其中三级模式由外到内分别是(　　)。

A. 模式、外模式、内模式　　B. 内模式、模式、外模式

C. 外模式、模式、内模式　　D. 模式、内模式、外模式

7. 关系数据模型的三个组成部分中,不包括(　　)。

A. 并发控制　　B. 数据结构

C. 完整性规则　　D. 数据操作

8. 用有向图结构表示实体类型以及实体类型之间联系的数据模型是(　　)。

A. 关系数据模型　　B. 层次数据模型

C. 网状数据模型　　D. 面向对象数据模型

9. 在E-R图中,主要成分包括实体和(　　)。

A. 结点、记录　　B. 属性、主码

C. 属性、联系　　D. 文件、关联

10. 学生社团可以接纳多名学生参加,但每个学生只能参加一个社团,从社团到学生之间的联系类型是(　　)。

A. 一对一　　B. 一对多　　C. 多对一　　D. 多对多

二、简答题

1. 简述数据库(DB)、数据库系统(DBS)、数据库管理系统(DBMS)的概念、区别和联系。

2. 目前通用的关系型数据库有哪些?(举三个例子)

3. 什么是数据与程序的逻辑独立性?什么是数据与程序的物理独立性?为什么数据库系统具有数据与程序的独立性?

三、应用题

1. 在一个医院中,有若干病人和若干医生,每个病人对应一个医疗记录,一个医生可以治疗若干个病人,一个病人对应一个医生。医生的属性有医生编号和医生姓名,病人的属性有病人编号和病人姓名,医疗记录的属性有记录编号和记录日期。试用E-R图加以描述,并标出实体标识符。

2. 某工厂生产若干产品,每种产品由不同的零件组成,有的零件可用在不同的产品上。

这些零件由不同的原材料制成，不同零件所用的材料可以相同。这些零件按所属的不同产品分别放在不同的仓库中，原材料按照类别放在若干不同仓库中。产品的属性有产品编号、产品数量和产品名；零件的属性有零件编号和零件名；材料的属性有材料编号和材料名；仓库的属性有仓库编号、仓库名和地点。请用 E-R 图画出此工厂产品、零件、材料、仓库的概念模型，并注明实体的属性、联系的类型以及实体标识符。

第 2 章

关系运算理论

2.1 关系数据结构

2.1.1 关系的定义

1. 域(Domain)

域是一组具有相同数据类型的值的集合。

例如,整数、实数、介于某个取值范围的日期。

2. 笛卡儿积(Cartesian Product)

给定一组域 $D_1, D_2, \cdots, D_n$,这些域中可以有相同的。$D_1, D_2, \cdots, D_n$的笛卡儿积为:

$$D_1 \times D_2 \times \cdots \times D_n = \{ (d_1, d_2, \cdots, d_n) \mid d_i \in D_i, i=1,2,\cdots,n \}$$

所有域的所有取值的任意组合,不能重复。

1) 元组(Tuple)

笛卡儿积中每一个元素$(d_1, d_2, \cdots, d_n)$叫作一个 n 元组(n-tuple)或简称元组。

2) 分量(Component)

笛卡儿积元素$(d_1, d_2, \cdots, d_n)$中的每一个值 d_i叫作一个分量。

3) 基数(Cardinal Number)

若 $D_i(i=1,2,\cdots,n)$为有限集,其基数为 $m_i(i=1,2,\cdots,n)$,则 $D_1 \times D_2 \times \cdots \times D_n$的基数 M 为:$M = \prod_{i=1}^{n} m_i$(即为元组的个数)。

4) 笛卡儿积的表示方法

笛卡儿积可表示为一个二维表。表中的每行对应一个元组,表中的每列对应一个域。

例如,给出如下 3 个域。

D_1:院系集合 DEPARTMENT={ 侦查系,刑事技术系 }

D_2:班级集合 CLASS={ 1 班,2 班 }

D_3:学生集合 STUDENT={ 张三,李四,王五 }

其中(侦查系,1 班,张三)、(侦查系,2 班,李四)等都是元组,侦查系、1 班、2 班、张三、李四等都是分量。

该笛卡儿积的基数为 $2\times2\times3=12$。即 $D_1\times D_2\times D_3$ 共有 12 个元组。这 12 个元组可列成一张二维表，如表 2.1 所示。

表 2.1　D_1、D_2、D_3 的笛卡儿积

DEPARTMENT	CLASS	STUDENT
侦查系	1 班	张三
侦查系	1 班	李四
侦查系	1 班	王五
侦查系	2 班	张三
侦查系	2 班	李四
侦查系	2 班	王五
刑事技术系	1 班	张三
刑事技术系	1 班	李四
刑事技术系	1 班	王五
刑事技术系	2 班	张三
刑事技术系	2 班	李四
刑事技术系	2 班	王五

3. 关系（Relation）

$D_1\times D_2\times\cdots\times D_n$ 的子集叫作在域 $D_1,D_2,\cdots,D_n$ 上的关系，表示为：

$$R(D_1,D_2,\cdots,D_n);$$

R 为关系名，n 为关系的目或度（Degree）。

1）元组

关系中的每个元素是关系中的元组，通常用 t 表示。

2）单元关系与二元关系

当 $n=1$ 时，称该关系为单元关系（Unary relation）。

当 $n=2$ 时，称该关系为二元关系（Binary relation）。

3）关系的表示

关系也是一个二维表，表的每行对应一个元组，表的每列对应一个域，如表 2.2 所示。其中表 2.2 是在表 2.1 的笛卡儿积中取出有意义的元组，形成的一个子集。

表 2.2　DCS 关系

DEPARTMENT	CLASS	STUDENT
侦查系	1 班	张三
刑事技术系	2 班	李四
刑事技术系	1 班	王五

4）属性

关系中不同列可以对应相同的域，为了加以区分，必须对每一个列起一个名字，称为属性(Attribute)。

5）码

(1) 候选码(Candidate key)。

若关系中的某一属性组的值能唯一地标识一个元组，而其任何真子集都不能再标识一个元组，则称该属性组为候选码，在最简单的情况下，候选码只包含一个属性。

例如，存在一个学生关系，包括学号、姓名、年龄、身份证号四个属性，其中学号可以唯一地标识一个学生元组，身份证号也可以唯一地标识一个学生元组，所以学号和身份证号可以作为学生关系的候选码。

(2) 全码(All-key)。

在最极端的情况下，关系的所有属性组是这个关系的候选码，称为全码(All-key)。

例如，存在一个音乐会关系，包括演奏者、音乐作品、观众三个属性，其中三个属性组合在一起才可以唯一地标识一个音乐会元组，所以关系的所有属性组是这个关系的候选码，即为全码。

(3) 主码(Primary key)。

若一个关系有多个候选码，则选定其中一个为主码或主键(Primary key)。

例如，在学生关系中，根据具体情况，学号可以更好地标识一个学生元组，所以学号作为学生关系的主码。当然身份证号也可以作为学生关系的主码。

(4) 主属性(Prime attribute)与非主属性(Non-key attribute)。

候选码的诸属性称为主属性(Prime attribute)。不包含在任何候选码中的属性称为非主属性(Non-key attribute)。

例如，在学生关系中，学号和身份证号为主属性，姓名和年龄为非主属性。

2.1.2 关系的性质

关系数据库中的关系必须具有下列一些性质。

(1) 列是同质的(Homogeneous)，即每一列中的分量是同一类型的数据，来自同一个域。

(2) 不同的列可出自同一个域，其中的每一列称为一个属性，不同的属性要给予不同的属性名。

(3) 列的顺序无所谓，即列的次序可以任意交换。在许多实际关系数据库产品中，增加新属性时，永远是插至最后一列。

(4) 任意两个元组的候选码不能相同。

(5) 行的顺序无所谓，行的次序可以任意交换。在许多实际关系数据库产品中，插入一个元组时永远插至最后一行。

（6）分量必须取原子值，即每一个分量都必须是不可分的数据项。这是规范条件中最基本的一条。

2.1.3　关系模式

关系模式是对关系的描述。关系模式在形式上可以表示为：

R(U,D,DOM,F)

其中 R 为关系名，U 为组成该关系的属性名集合，D 为属性组 U 中属性所来自的域，DOM 为属性向域的映像集合，F 为属性间的数据依赖关系集合。

1. 关系模式的表示

关系模式通常可以简记为：

R(U)

或

$R(A_1,A_2,\cdots,A_n)$

其中 R 为关系名，$A_1,A_2,\cdots,A_n$ 为属性名。而域名及属性向域的映像常常直接说明为属性的类型、长度。

2. 关系模式与关系

关系模式是对关系的描述，关系模式是静态的、稳定的；关系是关系模式在某一时刻的状态或内容，关系是动态的、随时间不断变化的；关系模式和关系往往统称为关系，通过上下文加以区别。

2.1.4　关系数据库

在关系模型中，实体以及实体间的联系都是用关系来表示的。在一个给定的应用领域中，所有实体及实体之间联系的关系的集合（也可以简单地理解为表的集合）构成一个关系数据库。

2.2　关系数据操作

2.2.1　关系的基本操作

关系模型中常用的关系操作包括查询操作和插入、删除、修改操作两大部分。

关系的查询表达能力很强，是关系操作中最主要的部分。查询操作可以分为选择（Select）、投影（Project）、连接（Join）、除（Divide）、并（Union）、交（Intersection）、差（Except）和笛卡儿积等。

其中，选择、投影、并、差、笛卡儿积是五种基本操作，其他操作是可以用基本操作来定义

和导出的。

2.2.2 关系操作的特点

关系操作的特点是集合操作方式，即操作的对象和结果都是集合。这种操作方式也称为一次一集合的方式，这里提到的“一个集合”也可以理解为“一个关系”或“一个二维表”。相应地，非关系数据模型的数据操作方式称为一次一记录的方式。

2.2.3 关系数据语言

(1) 关系数据语言按照完成的功能可分为三类：数据定义语言(DDL)、数据操纵语言(DML)和数据控制语言(DCL)。

① DDL 负责数据库的描述，提供一种数据定义机制，用来描述数据库的特征或数据的逻辑结构。

② DML 负责数据库的操作，提供一种数据处理操作的机制。DML 语言包括数据查询和数据的增加、删除、修改等功能，其中查询的表达方式是 DML 的主要部分。

③ DCL 负责控制数据库的完整性和安全性，提供一种检验完整性和保证安全的机制。

(2) 关系数据语言按照查询方式的不同可分为三类：关系代数语言(如 ISBL)、关系演算语言(如 APLHA)、具有关系代数与关系演算双重特点的语言(如 SQL)。

① 关系代数语言是用关系的集合运算来表达查询方式的语言。

② 关系演算语言是用谓词演算来表达查询方式的语言。关系演算又可按谓词变元的基本对象是元组变量还是域变量分为元组关系演算和域关系演算。

关系代数、元组关系演算和域关系演算三种语言在表达能力上是完全等价的。三者均是抽象的查询语言，这些抽象语言与具体的 DBMS 中实现的实际语言并不完全一样，但它们是实际 DBMS 软件产品中实现的具体查询语言的理论基础。

③ 另外还有一种介于关系代数语言和关系演算语言之间的结构化查询语言(Structured Query Language，SQL)。SQL 不仅具有丰富的查询功能，而且具有数据定义和数据控制功能，是集查询、DDL、DML 和 DCL 于一体的关系数据语言。它充分体现了关系数据语言的特点和优点，是关系数据库的标准语言。

2.3 关系的完整性

2.3.1 完整性约束的分类

关系模型中有三类完整性约束：实体完整性、参照完整性和用户定义的完整性。

(1) 实体完整性和参照完整性是关系模型必须满足的完整性约束条件，被称作是关系的两个不变性，应该由关系数据库 DBMS 自动支持。

（2）用户自定义的完整性是应用领域需要遵循的约束条件，体现了具体领域中的语义约束。

2.3.2　实体完整性

1. 实体完整性规则

若属性 A 是基本关系 R 的主属性，则属性 A 不能取空值。

2. 实体完整性的必要性

（1）实体完整性规则是针对基本关系而言的。一个基本表通常对应现实世界的一个实体集或一个多对多联系。

（2）现实世界中的实体和实体间的联系都是可区分的，即它们具有某种唯一性标识。相应地，关系模型中以主码作为唯一性标识。

（3）主码中的属性即主属性不能取空值。空值就是“不知道”或“无意义”的值。主属性取空值，就说明存在某个不可标识的实体，即存在不可区分的实体，这与第(2)点相矛盾。

3. 实体完整性规则规定：基本关系的所有主属性都不能取空值。

例如，学生实体中“学号”是主码，则“学号”不能取空值；课程实体中“课程号”是主码，则“课程号”不能取空值；学生选课关系——选修表(学号，课程号，成绩)中，“学号、课程号”是主码，则“学号”和“课程号”两个属性都不能取空值。

2.3.3　参照完整性

1. 关系间的引用

现实世界中的实体之间往往存在着某种联系，在关系模型中实体及实体间的联系都是用关系来描述的。这样就自然存在着关系与关系间的引用。

例如，学生、课程、学生与课程之间的多对多联系可以用如下三个关系表示：

学生(学号，姓名，性别，年龄，院系)
课程(课程号，课程名，学分)
选修(学号，课程号，成绩)

这三个关系(或三张表)之间存在着属性的引用，即选修关系引用了学生关系的主码“学号”和课程关系的主码“课程号”。同样，选修关系中的“学号”值必须是确实存在的学生的学号，即学生关系中有该学生的记录；选修关系中的“课程号”值也必须是确实存在的课程的课程号，即课程关系中有该课程的记录。换句话说，选修关系中某些属性的取值需要参照其他关系的属性取值。

2. 外码

设 F 是基本关系 R 的一个或一组属性，但不是关系 R 的码。如果 F 与基本关系 S 的主码 K_S 相对应，则称 F 是基本关系 R 的外码，基本关系 R 称为参照关系(Referencing Relation)，基本关系 S 称为被参照关系(Referenced Relation)或目标关系(Target Relation)。

在上例中，选修关系中的“学号”属性与学生关系的主码“学号”相对应；选修关系的“课

程号”属性与课程关系的主码“课程号”相对应，因此“学号”和“课程号”属性分别是选修关系的外码。这里学生关系和课程关系均为被参照关系，选修关系为参照关系。

3. 参照完整性规则

若属性(或属性组)F 是基本关系 R 的外码，它与基本关系 S 的主码 K_S 相对应(基本关系 R 和 S 不一定是不同的关系)，则对于 R 中每个元组在 F 上的值必须为：或取空值(F 的每个属性值均为空值)；或等于 S 中某个元组的主码值。

结合实例，按照参照完整性规则，“学号”和“课程号”属性也可以取两类值：空值或目标关系中已经存在的值。但由于“学号”和“课程号”是选修关系中的主属性，按照实体完整性规则，它们均不能取空值，并且选修关系中的“学号”和“课程号”属性实际上只能取相应被参照关系中已经存在的主码值。

2.3.4 用户定义完整性

实体完整性与参照完整性是由系统自动支持的，这是关系模型所要求的。除此之外，不同的关系数据库系统根据其应用环境的不同，往往需要一些特殊的约束条件，这就是用户定义的完整性约束条件。

(1) 用户定义完整性规则是针对某一具体关系数据库的约束条件，反映某一具体应用所涉及的数据必须满足的语义要求。

(2) 关系模型应提供定义和检验这类完整性的机制，以便用统一的系统的方法处理它们，而不要由应用程序承担这一功能。

例如，学生实体中，假如规定必须给出学生姓名，则必须使用用户定义的完整性约束要求学生姓名不能取空值；假如规定学生的年龄必须小于 35 岁，则必须使用用户定义的完整性约束，把年龄的取值范围规定在 0～35 岁之间等。

2.4 传统的集合运算

关系代数的运算对象和结果均为关系。关系代数用到的运算符包括集合运算符、专门的关系运算符、比较运算符和逻辑运算符，如表 2.3 所示。

表 2.3 关系代数运算符

运算符		含义	运算符		含义
集合运算符	∪	并	比较运算符	>	大于
	−	差		≥	大于等于
	∩	减		<	小于
	×	广义笛卡儿积		≤	小于等于
				=	等于
				≠	不等于

续表

运算符		含义	运算符		含义
专门的关系运算符	σ	选择	逻辑运算符	$\neg$	非
	Π	投影		$\wedge$	与
	$\bowtie$	连接		$\vee$	或
	$\div$	除			

关系代数的运算按运算符的不同可分为传统的集合运算和专门的关系运算两类。

传统的集合运算是二目运算，包括并、差、交、广义笛卡儿积四种运算。

设关系 R 和关系 S 具有相同的目 n（即两个关系都有 n 个属性），且相应的属性取自同一个域，t 是元组变量，$t \in R$ 表示 t 是 R 的一个元组。

2.4.1　并运算

例题 2.1　在校学生关系 R 和休学学生关系 S，其中关系 R 与关系 S 都有四个属性（学号，姓名，性别，状态），若要取得所有学生关系 T，则关系 T 由属于在校学生关系 R 和休学学生关系 S 的所有元组组成（即为集合并运算），并且得到的关系 T 仍然有四个属性（学号，姓名，性别，状态），如表 2.4～表 2.6 所示。

表 2.4　在校学生关系 *R*

学号	姓名	性别	状态
11432001	陈一	男	1
11432002	姚二	女	1
11432005	王五	男	1

表 2.5　休学学生关系 *S*

学号	姓名	性别	状态
11432003	张三	女	0
11432007	陈七	女	0

表 2.6　所有学生关系 *T*

学号	姓名	性别	状态
11432001	陈一	男	1
11432002	姚二	女	1
11432005	王五	男	1
11432003	张三	女	0
11432007	陈七	女	0

由此，关系 R 与关系 S 的并（Union）记作：

$$R \cup S = \{t \mid t \in R \lor t \in S\}$$

其结果关系仍为 n 目关系，由属于 R 或属于 S 的元组组成。

2.4.2 差运算

例题 2.2 有本店商品关系 R 和不合格商品关系 S，其中关系 R 与关系 S 都有三个属性（品牌，名称，厂家），若要找出本店内合格的商品关系 T，则关系 T 由属于本店商品关系 R 而不属于不合格商品关系 S 的元组组成（即为集合差运算），并且得到的关系 T 仍然有三个属性（品牌，名称，厂家），如表 2.7～表 2.9 所示。

表 2.7 本店商品关系 R

品牌	名称	厂家
A0001	婴儿奶粉	一厂
A0002	婴儿奶粉	二厂
B0001	绵白糖	一厂
B0002	绵白糖	四厂
C0001	牛肉干	三厂
D0001	巧克力	五厂
D0002	巧克力	六厂

表 2.8 不合格商品关系 S

品牌	名称	厂家
A0002	婴儿奶粉	二厂
B0001	绵白糖	一厂
B0003	绵白糖	二厂
D0001	巧克力	五厂
E0003	手指饼干	六厂

表 2.9 本店合格商品关系 T

品牌	名称	厂家
A0001	婴儿奶粉	一厂
B0002	绵白糖	四厂
C0001	牛肉干	三厂
D0002	巧克力	六厂

由此，关系 R 与关系 S 的差（Difference）记作：

$$R - S = \{t \mid t \in R \land t \notin S\}$$

其结果关系仍为 n 目关系，由属于 R 而不属于 S 的所有元组组成。

2.4.3　交运算

例题 2.3　在上例中，若要找出本店内不合格的商品关系 T，则关系 T 由既属于本店商品关系 R 又属于不合格商品关系 S 的元组组成（即为集合交运算），并且得到的关系 T 仍然有三个属性（品牌，名称，厂家），如表 2.10 所示。

表 2.10　本店不合格商品关系 T

品牌	名称	厂家
A0002	婴儿奶粉	二厂
B0001	绵白糖	一厂
D0001	巧克力	五厂

由此，关系 R 与关系 S 的交（Intersection）记作：

$$R \cap S = \{t \mid t \in R \wedge t \in S\}$$

其结果关系仍为 n 目关系，由既属于 R 又属于 S 的元组组成。关系的交运算可以用差运算来表示，即 $R \cap S = R-(R-S)$，或 $R \cap S = S-(S-R)$。

2.4.4　广义笛卡儿积

例题 2.4　现有学生关系 R 和必修课程关系 S，其中关系 R 有两个属性（学号、姓名），关系 S 有三个属性（课程号、课程名、学分），每个学生必须学习所有必修课程，要求形成选课关系 T。由学生选修课程形成的选课关系必须包括学生关系 R 的属性和必修课程关系 S 的属性，即关系 T 包括学号、姓名、课程号、课程名和学分五个属性，由于学生关系 R 中两个元组选修的课程都对应着必修课程关系 S 中的三个元组，所以，选课关系 T 共有六个元组，则得到的关系 T 由五个属性和六个元组组成（即为笛卡儿积运算），如表 2.11～表 2.13 所示。

表 2.11　学生关系 R

学号	姓名
11432001	陈一
11432002	姚二

表 2.12　必修课程关系 S

课程号	课程名	学分
c_1	大学英语	3
c_2	马克思主义基本原理	2
c_3	高等数学	3

表 2.13　选课关系 T

学号	姓名	课程号	课程名	学分
11432001	陈一	c_1	大学英语	3
11432001	陈一	c_2	马克思主义基本原理	2
11432001	陈一	c_3	高等数学	3
11432002	姚二	c_1	大学英语	3
11432002	姚二	c_2	马克思主义基本原理	2
11432002	姚二	c_3	高等数学	3

由此，两个分别为 n 目和 m 目的关系 R 和 S 的广义笛卡儿积是一个 $(n+m)$ 列的元组的集合。元组的前 n 列是关系 R 的一个元组，后 m 列是关系 S 的一个元组。若 R 有 k_1 个元组，S 有 k_2 个元组，则关系 R 和关系 S 的广义笛卡儿积有 $k_1 \times k_2$ 个元组。记作：

$$R \times S = \{\widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S\}$$

图 2.1(a)、图 2.1(b)分别为具有三个属性列的关系 R 和 S。图 2.1(c)为关系 R 与 S 的并。图 2.1(d)为关系 R 与 S 的交。图 2.1(e)为关系 R 和 S 的差。图 2.1(f)为关系 R 和 S 的广义笛卡儿积。

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_2	c_1

(a) R

A	B	C
a_1	b_2	c_2
a_1	b_3	c_2
a_2	b_2	c_1

(b) S

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_2	c_1
a_1	b_3	c_2

(c) $R \cup S$

A	B	C
a_1	b_2	c_2
a_2	b_2	c_1

(d) $R \cap S$

A	B	C
a_1	b_1	c_1

(e) $R-S$

A	B	C	A	B	C
a_1	b_1	c_1	a_1	b_2	c_2
a_1	b_1	c_1	a_1	b_3	c_2
a_1	b_1	c_1	a_2	b_2	c_1
a_1	b_2	c_2	a_1	b_2	c_2
a_1	b_2	c_2	a_1	b_3	c_2
a_1	b_2	c_2	a_2	b_2	c_1
a_2	b_2	c_1	a_1	b_2	c_2
a_2	b_2	c_1	a_1	b_3	c_2
a_2	b_2	c_1	a_2	b_2	c_1

(f) $R \times S$

图 2.1　传统的集合运算

2.5　专门的关系运算

专门的关系运算包括选择、投影、连接、除运算等。下面给出这些关系运算的定义。

2.5.1　选择运算

设有一个学生—课程数据库。学生关系包括学号、姓名、性别、年龄和院系五个属性，课程关系包括课程号、课程名和学分三个属性，选修关系包括学号、课程号和成绩三个属性。关系模式表示如下：

```
student(sno,sname,sex,age,dept)
course(cno,cname,credit)
sc(sno,cno,grade)
```

三个关系中的具体数据如表 2.14～表 2.16 所示。下面的许多例子都将对这三个关系进行运算。

表 2.14　学生关系 student

sno	sname	sex	age	dept
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系

表 2.15　课程关系 course

cno	cname	credit
c1	大学英语	3
c2	马克思主义基本原理	2
c3	高等数学	3
c4	证据学	2

表 2.16　选课关系 sc

sno	cno	grade
11432001	c1	75
11432001	c2	95
11432001	c3	82
11432001	c4	88

续表

sno	cno	grade
11432002	c1	89
11432002	c3	61
11432003	c1	72
11432003	c2	45
11432003	c3	66
11432004	c2	85
11432004	c3	97

例题 2.5 查询侦查系全体学生的信息。由 student 关系中满足 dept=“侦查系”这一条件的元组组成(即为选择运算)。结果如表 2.17 所示。

表 2.17 侦查系全体学生

sno	sname	sex	age	dept
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系

由此可见,选择运算实际上是从关系 R 中选取使逻辑表达式值为真的元组。这是从行的角度进行的运算,如图 2.2 所示。

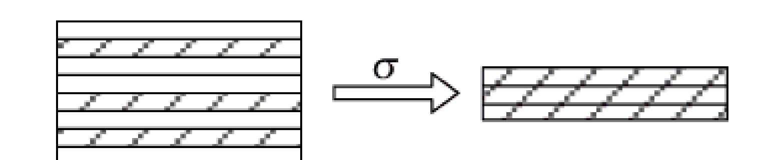


图 2.2 选择操作

选择又称为限制(Restriction)。它是在关系 R 中选择满足给定条件的诸元组,记作:

$$\sigma_F(R)=\{t \mid t \in R \wedge F(t)= \text{'真'}\}$$

其中 F 表示选择条件,它是一个逻辑表达式,取逻辑值“真”或“假”。逻辑表达式 F 由逻辑运算符 ∨、∧、¬ 连接各算术表达式组成。算术表达式的基本形式为:

$$X_1\ \theta\ Y_1$$

其中 θ 表示比较运算符,它可以是>、≥、<、≤、=或≠。X_1、Y_1 等是属性名,或为常量,或为简单函数;属性名也可以用它的序号来代替。

例题 2.6 查询年龄小于 20 岁的学生。

$$\sigma_{\text{Age}<20}(\text{student})\ 或\ \sigma_{4<20}(\text{student})$$

结果如表 2.18 所示。

表 2.18　年龄小于 20 岁的学生

sno	sname	sex	age	dept
11432001	陈一	男	17	侦查系
11432003	张三	女	19	侦查系

2.5.2　投影运算

例题 2.7　查询学生的学号和姓名。由 student 关系中的学号属性列和姓名属性列组成的新关系(即为投影运算)。结果如表 2.19 所示。

表 2.19　学号和姓名

sno	sname
11432001	陈一
11432002	姚二
11432003	张三
11432004	李四

说明：形成的新关系不仅取消了原关系中的某些列，而且还可能取消某些元组，因为取消了某些属性列后，就可能出现重复行，应取消这些完全相同的行。

由此可见，投影操作是从列的角度进行的运算，如图 2.3 所示。

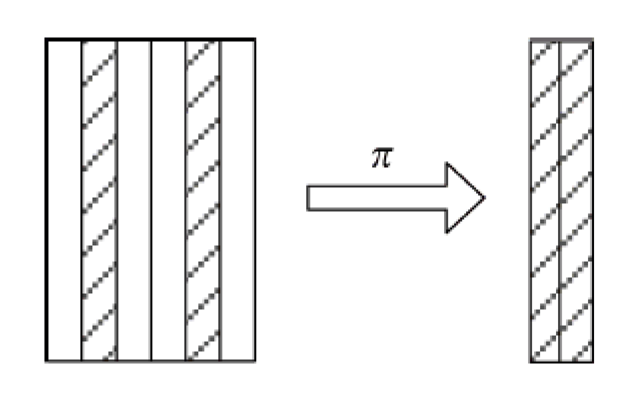


图 2.3　投影操作

关系 R 上的投影是从 R 中选择出若干属性列组成新的关系。记作：

$$\pi_A(R)=\{t[A]\mid t\in R\}$$

其中 A 为 R 中的属性列。

例题 2.8　查询学生关系 student 中都有哪些院系，即查询关系 student 在院系属性上的投影。

$$\pi_{\text{Dept}}(\text{student})$$

结果如表 2.20 所示。

说明：student 关系原来有四个元组，而投影结果取消了重复的侦查系元组，因此只有两个元组。

表 2.20　学生所在院系

dept
侦查系
刑事技术系

2.5.3 连接运算

连接也称为 θ 连接。它是从两个关系的笛卡儿积中选取属性间满足一定条件的元组。记作：

$$R \underset{A\theta B}{\bowtie} S = \{\widehat{t_r t_s} \mid t_r \in R \land t_s \in S \land t_r[A]\theta t_s[B]\}$$

其中 A 和 B 分别为 R 和 S 上度数相等且可比的属性组。θ 是比较运算符。连接运算从 R 和 S 的广义笛卡儿积 $R \bowtie S$ 中选取(R 关系)在 A 属性组上的值与(S 关系)在 B 属性组上值满足比较关系 θ 的元组。

连接运算中有两种最为重要也最为常用的连接：一种是等值连接(Equal-Join)，一种是自然连接(Natural-Join)。

θ 为"＝"的连接运算称为等值连接。它是从关系 R 与 S 的广义笛卡儿积中选取 A、B 属性值相等的那些元组，即等值连接为：

$$R \underset{A=B}{\bowtie} S = \{\widehat{t_r t_s} \mid t_r \in R \land t_s \in S \land t_r[A]\theta t_s[B]\}$$

自然连接(Natural-join)是一种特殊的等值连接，它要求两个关系中进行比较的分量必须是相同的属性组，并且在结果中把重复的属性列去掉。即若 R 和 S 具有相同的属性组 B，则自然连接可记作：

$$R \bowtie S = \{\widehat{t_r t_s} \mid t_r \in R \land t_s \in S \land t_r[A]\theta t_s[B]\}$$

一般的连接操作是从行的角度进行运算。但自然连接还需要取消重复列，所以是同时从行和列的角度进行运算，如图 2.4 所示。

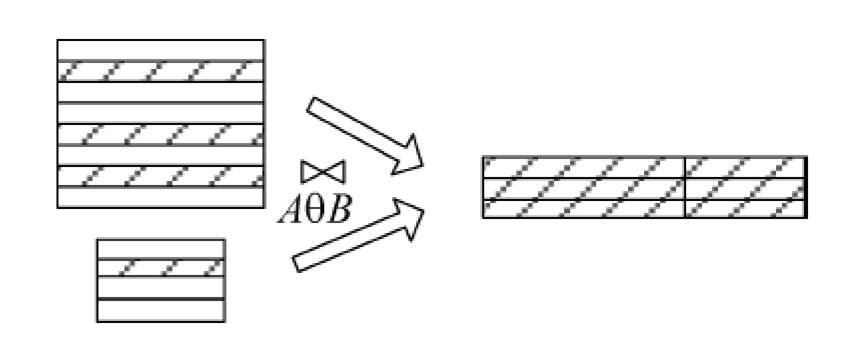


图 2.4 连接操作

两个关系 R 和 S 在做自然连接时，选择两个关系在公共属性上值相等的元组构成新的关系。此时，关系 R 中某些元组可能在 S 中不存在公共属性上值相等的元组，从而造成 R 中这些元组在操作时被舍弃了，同样，S 中某些元组也可能被舍弃，例如，在图 2.5 的自然连接中，R 中的第 4 个元组、S 中的第 5 个元组被舍掉了。

如果把舍弃的元组也保存在结果关系中，而在其他属性上填空值(NULL)，那么这种连接就叫作外连接(Outer Join)，外连接可记作 R ⟗ S。如果只把左边关系 R 中要舍弃的元组保留就叫作左外连接(Left Outer Join)，左外连接记作 R ⟕ S。如果只把右边关系 S 中要舍弃的元组保留就叫作右外连接(Right Outer Join)，右外连接记作 R ⟖ S。

图 2.5(a)和(b)分别为关系 R 和关系 S，图 2.5(c)为 $R \underset{C<E}{\bowtie} S$ 的结果，图 2.5(d)为等值

连接 $R \underset{R.B=S.B}{\bowtie} S$ 的结果。图 2.5(e)为自然连接 $R \bowtie S$ 的结果，图 2.5(f)为外连接 $R ⟗ S$ 的结果。图 2.5(g)为左外连接 $R ⟕ S$ 的结果。图 2.5(h)为右外连接 $R ⟖ S$ 的结果。

R

A	B	C
a_1	b_1	5
a_1	b_2	6
a_2	b_3	8
a_2	b_4	12

(a) 关系 R

S

B	E
b_1	3
b_2	7
b_3	10
b_3	2
b_5	2

(b) 关系 S

$R \underset{C<E}{\bowtie} S$

A	R.B	C	S.B	E
a_1	b_1	5	b_2	7
a_1	b_1	5	b_3	10
a_1	b_2	6	b_2	7
a_1	b_2	6	b_3	10
a_2	b_3	8	b_3	10

(c) 一般连接

A	R.B	C	S.B	E
a_1	b_1	5	b_1	3
a_1	b_2	6	b_2	7
a_2	b_3	8	b_3	10
a_2	b_3	8	b_3	2

(d) 等值连接

A	B	C	E
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2

(e) 自然连接

A	B	C	E
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2
a_2	b_4	12	NULL
NULL	b_5	NULL	2

(f) 外连接

A	B	C	E
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2
a_2	b_4	12	NULL

(g) 左外连接

A	B	C	E
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2
NULL	b_5	NULL	2

(h) 右外连接

图 2.5　连接运算举例

2.5.4　除运算

给定关系 $R(X,Y)$ 和 $S(Y,Z)$，其中 X、Y、Z 为属性组。R 中的 Y 与 S 中的 Y 可以有不同的属性名，但必须出自相同的域集。R 与 S 的除运算得到一个新的关系 $P(X)$，P 是 R 中满足下列条件的元组在 X 属性列上的投影：元组在 X 上分量值 x 的像集 Y_x 包含 S 在 Y 上

投影的集合。记作：

$$R \div S = \{t_r[x] \mid t_r \in R \wedge \pi_y(S) \subseteq Y_x\}$$

其中 Y_x 为 x 在 R 中的像集，$x = t_r[X]$。

除操作是同时从行和列角度进行运算，如图 2.6 所示。

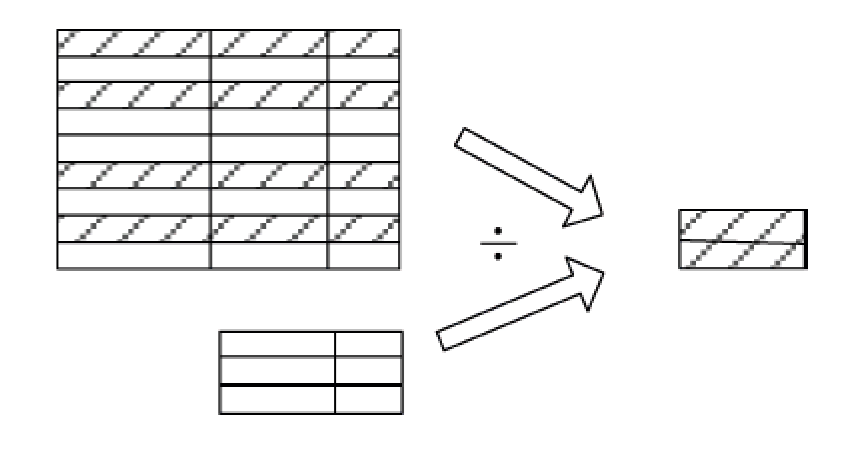


图 2.6 除操作

除运算如图 2.7 所示。设关系 R、S 分别如图 2.7 中的(a)和(b)所示，$R \div S$ 的结果如图 2.7(c)所示。

在关系 R 中，A 可以取四个值$\{a_1, a_2, a_3, a_4\}$，其中：

a_1的像集为$\{(b_1, c_2), (b_2, c_3), (b_2, c_1)\}$

a_2的像集为$\{(b_3, c_7), (b_2, c_3)\}$

a_3的像集为$\{(b_4, c_6)\}$

a_4的像集为$\{(b_6, c_6)\}$

S 在(B, C)上的投影为$\{(b_1, c_2), (b_2, c_1), (b_2, c_3)\}$

显然只有 a_1的像集(B, C)包含了 S 在(B, C)属性组上的投影，所以 $R \div S = \{a_1\}$。

A	B	C
a_1	b_1	c_2
a_2	b_3	c_7
a_3	b_4	c_6
a_1	b_2	c_3
a_4	b_6	c_6
a_2	b_2	c_3
a_1	b_2	c_1

(a) R

B	C	D
b_1	c_2	d_1
b_2	c_1	d_1
b_2	c_3	d_2

(b) S

A
a_1

(c) $R \div S$

图 2.7 除运算

例题 2.9 已知学生选课关系 R，课程表 S，要找出选课关系中选修所有课程的学生的学号。指定的课程号和对应课程名构成的关系记为 S，则该问题可以用 $R \div S$ 表示，如表 2.21～表 2.23 所示。

表 2.21　选课关系 *R*

学号	课程号
11432001	$c1$
11432001	$c2$
11432001	$c3$
11432001	$c4$
11432002	$c1$
11432002	$c3$
11432003	$c1$
11432003	$c2$
11432003	$c3$
11432004	$c2$
11432004	$c3$

表 2.22　课程关系 *S*

课程号	课程名
$c1$	大学英语
$c2$	马克思主义基本原理
$c3$	高等数学
$c4$	证据学

表 2.23　关系 $R \div S$

学号
11432001

2.6　综合实例

以学生—课程数据库为例,给出几个综合应用多种关系代数运算进行查询的例子。

学生关系：student(sno,sname,age,sex,dept)
课程关系：course(cno,cname,credit)
选课关系：sc(sno,cno,grade)

其中,学生关系中学号 sno 为主键,课程关系中课程号 cno 为主键,选课关系中学号 sno 和课程号 cno 是外键,分别参照学生和课程关系中的主键 sno 和 cno,如图 2.8 所示。

(1) 查询选修课程号为"$c1$"的学生学号和成绩。

(2) 查询学分为 3 的课程的课程号和课程名。

(3) 查询年龄小于 20 岁的女学生的学号与姓名。

(4) 查询选修学分为 3 的课程的学生学号。

(5) 查询学号为 11432003 学生所学课程的课程名与学分。

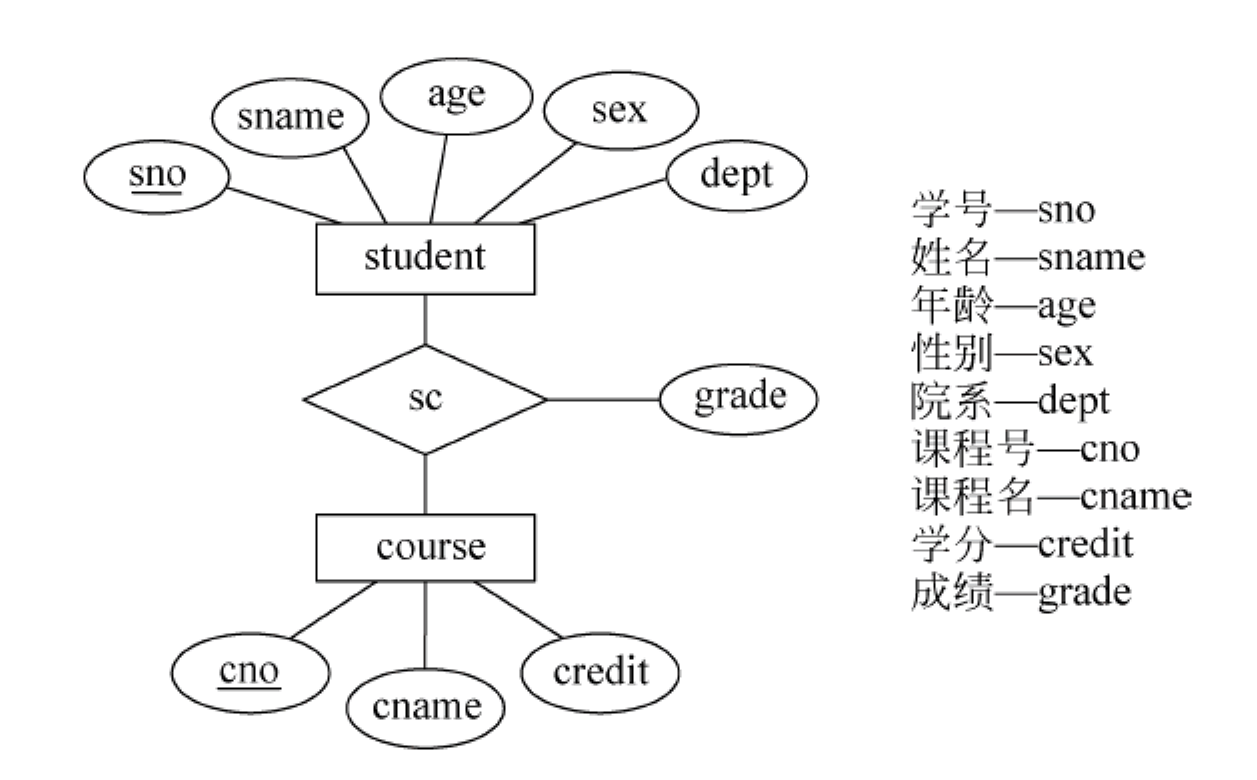


图 2.8 学生选课关系图

(6) 查询学习课程号为 $c2$ 的学生学号和姓名。

(7) 查询选修课程号为 $c1$ 或 $c3$ 的学生学号。

(8) 查询至少选修课程号为 $c1$ 和 $c3$ 的学生学号。

(9) 查询至少选修两门课程的学生学号。

(10) 查询选修课程名为“高等数学”的学生学号和姓名。

(11) 查询至少选修一门学分为 2 的课程的男学生的姓名。

(12) 查询“张三”不学的课程的课程号。

(13) 查询学习全部课程的学生姓名。

(14) 查询所学课程包含学生 11432004 所学课程的学生学号。

实例解析：

(1) 查询的结果是学生的学号和成绩，所以要做投影运算，从 sc 关系中投影学号 sno 和成绩 grade 两列。另外查询的是学习课程号为 $c1$ 的学生，必须做选择运算。综合选择和投影操作，先对关系 SC 执行选择操作，然后执行投影操作。关系代数表达式如下：

$$\pi_{\text{sno, grade}}(\sigma_{\text{cno='c1'}}(\text{sc}))$$

(2) 因为课程号、课程名和学分是 course 关系中的属性，所以只需要从 course 关系中用选择和投影操作即可实现。关系代数表达式如下：

$$\pi_{\text{cno, cname}}(\sigma_{\text{credit}=3}(\text{course}))$$

(3) 因为学号、姓名、性别和年龄都是 student 关系中的属性，所以只需要从 student 关系中进行选择和投影操作即可实现。关系代数表达式如下：

$$\pi_{\text{sno, sname}}(\sigma_{\text{age}<20 \wedge \text{sex='女'}}(\text{student}))$$

(4) 因为查询学习学分为 3 的课程的学生信息，所以必须做选择运算。因为查询的结果是学号，所以必须做投影运算。由于这个查询涉及关系 course 中的学分 credit 和关系 sc 中的学生号 sno，因此先将关系 course 和 sc 做自然连接运算，再对连接的结果做选择，选择

学习学分为 3 的课程的学生，最后再对结果进行投影操作。关系代数表达式如下：

$$\pi_{sno}(\sigma_{credit=3}(course \bowtie sc))$$

（5）因为学分在 course 关系中，学生选修信息在 sc 关系中，所以必须先对 course 和 sc 关系做自然连接运算，再对连接的结果按学号为 11432003 进行选择，对选择后的结果按课程名和学分进行投影。关系代数表达式如下：

$$\pi_{cname, credit}(\sigma_{sno='11432003'}(sc \bowtie course))$$

（6）因为查询学习课程号为 $c2$ 的学生信息，所以必须做选择运算。因为查询的结果是学号和姓名，所以必须做投影运算。由于这个查询涉及关系 student 中的学号 sno、学生姓名 sname 和关系 sc 中的课程号 cno，因此先将关系 student 和 sc 做自然连接运算，再对连接的结果做选择，选择学习课程号为 $c2$ 的学生，最后再对结果进行投影操作。关系代数表达式如下：

$$\pi_{sno, sname}(\sigma_{cno='c2'}(student \bowtie sc))$$

（7）因为查询的内容是学生学号，并且是选修课程号为 $c1$ 或 $c3$ 的学生，由于只涉及 sc 关系，所以需要对 sc 关系进行投影和选择操作。关系代数表达式如下：

$$\pi_{sno}(\sigma_{cno='c1' \vee cno='c3'}(sc))$$

（8）因为查询同时选修课程号为 $c1$ 和 $c3$ 的学生信息，所以将 sc 关系自身做笛卡儿积运算，运算后的结果如表 2.24 所示。要求同时选修 $c1$ 和 $c3$ 课程，表示笛卡儿积的结果中第 2 列值必须为 $c1$，第 5 列值必须为 $c3$。并且必须是同一个学生，所以笛卡儿积中第 1 列的值必须与第 4 列的值相等。最后结果进行投影显示学号 sno。表达式中采用属性序号代替属性名。关系代数表达式如下：

$$\pi_1(\sigma_{1=4 \wedge 2='c1' \wedge 5='c3'}(sc \times sc))$$

表 2.24　sc 关系自身做笛卡儿积的结果

1	2	3	4	5	6
sno	cno	grade	sno	cno	grade

（9）首先对 sc 关系进行自身的笛卡儿积运算，得到如表 2.24 所示的结果。至少选修两门课，表示笛卡儿积的结果中第 2 列的值和第 5 列的值必须不同。同一个学生选修多门课，则笛卡儿积的结果中第 1 列的学号 sno 与第 4 列的 sno 必须相同。最后按学号 sno 进行投影，关系代数表达式如下：

$$\pi_1(\sigma_{1=4 \wedge 2 \neq 5}(sc \times sc))$$

（10）因为课程名只有在 course 关系中才能找到，而学生姓名只有在 student 关系中才能找到，选修“高等数学”课程的选修关系只有在 sc 关系中才能找到，所以必须对三个关系（student、sc、course）做自然连接运算，再对连接的结果进行选择操作，最后对结果进行投影

操作。关系代数表达式如下：

$$\pi_{sno,\ sname}(\sigma_{cname='高等数学'}(student \bowtie sc \bowtie course))$$

(11) 因为要查询学生的姓名，所以需要从 student 关系中获取信息。因为涉及学分为 2 的课程，所以需要从 course 关系中获取信息。又因为涉及选修学分为 2 的课程，所以需要从 sc 关系中获取信息。所以查询涉及 student、sc、course 三个关系，先对这三个关系做自然连接运算，再对连接的结果进行选择操作，最后对结果进行投影操作。关系代数表达式如下：

$$\pi_{sname}(\sigma_{sex='男' \wedge credit=2}(student \bowtie sc \bowtie course))$$

(12) 首先用投影操作从 Course 关系中选取所有的课程号，然后求出“张三”选修的所有课程号，最后对这两个结果集进行集合差运算，即可求得“张三”同学不学的课程号。求“张三”选修的所有课程号时，因为只有 student 关系才有学生的姓名，只有 sc 关系中才有选修信息，所以涉及 student 和 sc 关系，先对这两个关系做自然连接运算，再对连接的结果进行选择操作，最后对选择的结果进行投影操作。关系代数表达式如下：

$$\pi_{cno}(course) - \pi_{cno}(\sigma_{sname='张三'}(student \bowtie sc))$$

(13) 首先用 $\pi_{sno,\ cno}(sc)$投影操作从 sc 关系中获取学生选课情况，然后用 $\pi_{cno}(course)$投影操作从 course 关系中获取全部课程信息，学习全部课程的学生学号可以用除操作 $\pi_{sno,\ cno}(sc) \div \pi_{cno}(course)$实现，操作的结果是学习全部课程的学生的学号 sno。由于需要取的结果是学生姓名，所以必须根据得到的学号从 student 关系中取出对应的姓名。可以将关系 student 与除操作的结果进行自然连接操作，得到的结果按学生姓名进行投影即可。关系代数表达式如下：

$$\pi_{sname}(student \bowtie (\pi_{sno,\ cno}(sc) \div \pi_{cno}(course)))$$

(14) 首先用 $\pi_{sno,\ cno}(sc)$操作从 sc 关系中获取全部学生的选课情况，然后用 $\pi_{cno}(\sigma_{sno='11432004'}(sc))$操作从 sc 关系中选取学生 11432004 所学的全部课程。求包含学生 11432004 所学的全部课程的学生学号，可以用除操作实现。关系代数表达式如下：

$$\pi_{sno,\ cno}(sc) \div \pi_{cno}(\sigma_{sno='11432004'}(sc))$$

2.7 本章小结

本章首先介绍了关系数据结构的基本概念、数学定义。包括关系的定义、关系的 6 个性质、关系模式的表示和关系数据库的相关概念。

其次介绍了关系模型由数据结构、关系操作、关系完整性约束三部分组成。关系模型必须遵循实体完整性规则、参照完整性规则和用户定义的完整性规则。

最后介绍了关系代数传统的集合运算和专门的关系运算。其中并、差、笛卡儿积、投影、选择这五种运算为基本的运算，它们组成关系代数完备的运算集。其他三种运算，即交、连接、除运算都可以用这五种基本运算组合而成。

2.8　课后习题

一、选择题

1. 设关系 R 的属性列数以及元组个数分别为 2 和 3，关系 S 的属性列数以及元组个数分别为 3 和 5，关系 T 是 R 和 S 的笛卡儿积，即 $T=R\times S$，则关系 T 的属性列数以及元组个数为(　　)。

A. 5，8　　B. 5，15　　C. 6，8　　D. 6，15

2. 在基本的关系中，下列说法正确的是(　　)。

A. 行列顺序有关　　B. 属性名允许重名

C. 任意两个元组不允许重复　　D. 列是非同质的

3. 关系中的“主码”不允许取空值是指(　　)规则。

A. 数据完整性　　B. 引用完整性

C. 用户自定义完整性　　D. 实体完整性

4. 在关系模式 R 中，若属性或属性组 X 不是关系 R 的关键字，但 X 是其他关系模式的关键字，则称 X 为关系 R 的(　　)。

A. 主键　　B. 超键　　C. 外键　　D. 候选键

5. 关系操作的对象和结果都是(　　)。

A. 元组　　B. 集合　　C. 属性　　D. 以上均不对

6. 关系操作中的五个基本操作是(　　)。

A. 并、差、交、选择、笛卡儿积

B. 并、差、交、投影、选择

C. 并、差、交、选择、除

D. 并、差、笛卡儿积、投影、选择

7. 参加差运算的两个关系(　　)。

A. 属性个数可以不相同　　B. 属性名必须相同

C. 属性个数必须相同　　D. 一个关系包含另一个关系的属性

8. 设两个关系 R 和 S，分别包含 15 和 10 个元组，则在 $R\cup S$、$R\cap S$、$R-S$ 中，可能出现的元组数目情况是(　　)。

A. 18，7，8　　B. 15，5，10　　C. 21，11，4　　D. 25，15，0

9. 取出关系的某些列，并取消重复元组的关系代数运算称为(　　)。

A. 交运算　　B. 选择运算　　C. 连接运算　　D. 投影运算

10. 在连接运算中如果两个关系中进行比较的分量必须是相同的属性组，那么这个连接是(　　)。

A. 有条件的连接　B. 等值连接　C. 自然连接　D. 完全连接

二、简答题

1. 存在关系 R 和 S，$R\cap S$ 的运算可用哪个等价的表达式来替换？

2. 假设关系 R 和 S 是具有相同属性列的关系，它们分别有 m 个元组和 n 个元组（假设 $m>n$），分别写出下列表达式中可能的最小和最大的元组数量。

$$R\cap S、R\cup S、R-S、R\bowtie S$$

3. 关系模型的完整性规则有哪几类？简述各完整性的含义。

4. 试述等值连接与自然连接的区别和联系。

三、应用题

某数据库中包含商店信息、商品信息和销售信息三张基本表。

商店信息表：shop(sid，sname，address)，表中属性列依次是商店编号、商店名称和地址。

商品信息表：commodity(cid，cname，spec，price)，表中属性列依次是商品编号、商品名称、规格和单价。

销售信息表：sale(sid，cid，quantity)，表中属性列依次是商店编号、商品编号和销售量。

试用关系代数运算完成如下检索：

(1) 查询商店名称为“旺达超市”所在的地址。

(2) 查询单价在 100～150 元的商品编号。

(3) 查询销售商品名称为“旺仔牛奶”的商店名称。

(4) 查询没有销售商品单价超过 5000 元的商店编号。

(5) 查询至少销售商品编号为 C1 和 C2 的商店编号。

第 3 章

关系数据库标准语言 SQL

3.1 SQL 概述

3.1.1 SQL 简介

结构化查询语言(Structured Query Language,SQL)是一种用于和关系数据库进行交互通信的计算机语言,用于存取数据以及查询、更新和管理关系数据库系统。同时它也是数据库脚本文件的扩展名。

SQL 最早是 1974 年由 Boyce 和 Chamberlin 提出,并作为 IBM 公司研制的关系数据库管理系统原型 System R 的一部分付诸实施。它功能丰富,不仅具有数据定义、数据操纵、数据控制功能,还有着强大的查询功能,而且语言简洁,容易学习,易于使用。现在 SQL 已经成为关系数据库的国际标准语言,各个数据库厂家纷纷推出各自的 SQL 软件或与 SQL 的接口软件。这就使大多数数据库均用 SQL 作为共同的数据存取语言和标准接口,使不同数据库系统之间的相互操作有了共同的基础,其意义是十分重大的。

SQL 语言的应用更加广泛,Oracle、Sybase、Informix、Ingres、DB2、SQL Server、Rdb 等大型数据库管理系统都实现了 SQL 语言;DBase、Foxpro、Access 等 PC 数据库管理系统部分实现了 SQL 语言;可以在 HTML(Hypertext Markup Language,超文本标记语言)中嵌入 SQL 语句,通过 WWW 访问数据库;在 VC、VB、Delphi、PB 中也可嵌入 SQL 语句。目前,很多数据库产品都对 SQL 语句进行再开发与扩展,如 Oracle 提供的 PL/SQL (Procedure Language and SQL)就是对 SQL 的一种扩展。

3.1.2 SQL 发展历程

SQL 随着数据库技术的发展而不断更新、丰富,下面介绍 SQL 的发展历程。

(1) 1970 年:E. F. Codd 发表了关系数据库理论(relational database theory)。

(2) 1974—1979 年:IBM 以 Codd 的理论为基础开发了 Sequel,并重命名为“结构化查询语言”。

(3) 1979 年:Oracle 发布了商业版结构化查询语言。

(4) 1981—1984 年:出现了其他商业版本,分别为 IBM(DB2)、Data General 和

Relational Technology(INGRES)。

(5) 1986 年：美国 ANSI 采用 SQL 作为关系数据库管理系统的标准语言，后为国际标准化组织(ISO)采纳为国际标准。

(6) 1989 年：结构化查询语言/89 增加了引用完整性(referential integrity)。

(7) 1992 年：结构化查询语言/92 被数据库管理系统(DBMS)生产商广泛接受。

(8) 1997 年以后：成为动态网站(dynamic web content)的后台支持。

(9) 2003 年：结构化查询语言/2003 包含了 XML 相关内容，自动生成列值(column values)。

(10) 2006 年：结构化查询语言/2006 定义了结构化查询语言与 XML(包含 XQuery)的关联应用。

(11) 2006 年：Sun 公司将以结构化查询语言基础的数据库管理系统嵌入 Java 6.0。

3.1.3 SQL 特点

SQL 语言是一个综合的、通用的、功能极强的、简学易用的语言，所以能够被用户和业界广泛接受，并成为国际标准。其主要特点如下：

1. 综合统一

SQL 语言集数据定义语言 DDL、数据操纵语言 DML、数据查询语言 DQL、数据控制语言 DCL 的功能于一体，语言风格统一，可以独立完成数据库生命周期中的全部活动，包括定义关系模式、插入数据、建立数据库、查询、更新、维护、数据库重构、数据库安全性控制等一系列操作要求，这些为数据库应用系统的开发提供了良好的环境。

SQL 语言的核心内容包括如下数据语言：

(1) 数据定义语言(Data Definition Language，DDL)，用于定义数据库的逻辑结构，包括基本表、视图及索引的定义。

(2) 数据操纵语言(Data Manipulation Language，DML)，用于对关系模式中的具体数据进行增、删、改等操作。

(3) 数据查询语言(Data Query Language，DQL)，用于实现各种不同的数据查询。

(4) 数据控制语言(Data Control Language，DCL)，用于数据访问权限的控制。

2. 高度非过程化

SQL 语言是非过程化的语言，用户只需提出“做什么”，而不必指明“怎么做”，也不需要了解存取路径的选择，SQL 语言就可以将要求交给系统，自动完成全部工作。这不但大大减轻了用户负担，而且有利于提高数据独立性。

3. 面向集合的操作方式

非关系数据模型采用的是面向记录的操作方式，操作对象是一条记录。而 SQL 语言采用集合操作方式，不仅操作对象、查找结果可以是元组的集合，而且一次插入、删除、更新操作的对象也可以是元组的集合。

4. 以同一种语法结构提供两种使用方式

SQL 语言既是独立的语言，又是嵌入式语言。作为独立的语言，它能够独立地用于联机交互的使用方式，用户可以在终端键盘上直接键入 SQL 命令对数据库进行操作。作为嵌入式语言，SQL 语句能够嵌入到高级语言(例如 C、Cobol、Fortran、C++、Java 等)程序中，供程序员设计程序时使用。现在很多数据库应用开发工具，都将 SQL 语言直接融入到自身的语言中，使用起来更加方便。尽管 SQL 的使用方式不同，但 SQL 语言的语法基本上是一致的。这种统一的语法结构提供两种不同的使用方式，为用户提供了极大的灵活性与方便性。

5. 语言简洁，易学易用

SQL 语言功能极强，但其语言十分简洁，完成数据定义、数据操纵、数据控制的核心功能只用了 9 个动词：CREATE、DROP、ALTER、SELECT、INSERT、UPDATE、DELETE、GRANT、REVOKE，如表 3.1 所示。而且 SQL 语言语法简单，接近英语口语，因此易学易用。

表 3.1　SQL 语言的动词

SQL 功能	命令动词
数据查询	SELECT
数据定义	CREATE、DROP、ALTER
数据操纵	INSERT、UPDATE、DELETE
数据控制	GRANT、REVOKE

3.2　数据定义

通过 SQL 语言的数据定义功能，可以完成基本表、视图、索引的创建、修改和删除。但 SQL 不提倡修改视图和索引的定义，如果想修改视图和索引的定义，只能先将它们删除，然后再重建。SQL 常用的数据定义语句，如表 3.2 所示。

表 3.2　SQL 的数据定义语句

操作对象	操作方式		
	创建	删除	修改
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	—
索引	CREATE INDEX	DROP INDEX	—

由于视图的定义与查询操作有关，本节只介绍基本表和索引的数据定义。

3.2.1 基本数据类型

由于基本表的每个属性列都有自己的数据类型，所以首先介绍一下 SQL 所支持的数据类型。各个厂家的 SQL 所支持的数据类型不完全一致，这里只介绍 SQL-99 规定的主要数据类型。

1. 数值型

(1) INTEGER 定义数据类型为整数类型，它的精度(总有效位)由执行机构确定。INTEGER 可简写成 INT。

(2) SMALLINT 定义数据类型为短整数类型，它的精度由执行机构确定。

(3) NUMERIC(p,s)定义数据类型为数值型，并给定精度 p(总的有效位，不包含符号位及小数点)或标度 s(十进制小数点右边的位数)。

(4) FLOAT(p)定义数据类型为浮点数值型，p 为指定的精度。

(5) REAL 定义数据类型为浮点数值型，它的精度由执行机构确定。

(6) DOUBLE PRECISION 定义数据类型为双精度浮点类型，它的精度由执行机构确定。

2. 字符类型

(1) CHAR(n)定义指定长度的字符串，n 为字符数的固定长度。

(2) VARCHAR(n)定义可变长度的字符串，其最大长度为 n，n 不可省略。

3. 位串型

(1) BIT(n)定义数据类型为二进制位串，其长度为 n。

(2) BIT VARYING(n)定义可变长度的二进制位串，其最大长度为 n，n 不可省略。

4. 时间型

(1) DATE 用于定义日期，包含年、月、日，格式为 YYYY-MM-DD。

(2) TIME 用于定义时间，包含时、分、秒，其格式为 HH:MM:SS。

5. 布尔型

BOOLEAN 定义布尔类型，其值可以是 TRUE(真)、FALSE(假)。

对于数值型数据，可以执行算术运算和比较运算，但对其他类型数据，只可以执行比较运算，不能执行算术运算。这里只介绍了常用的一些数据类型，许多 SQL 产品还扩充了其他一些数据类型，用户在实际使用中应查阅数据库系统的参考手册。

3.2.2 约束条件

在 SQL 语言中，约束是一些规则，约束在数据库中不占存储空间。根据约束所完成的功能不同，表达完整性约束的规则有主键约束、外键约束、属性约束三类。

1. 主键约束(PRIMARY KEY)

主键约束体现了实体完整性。要求某一列的值既不能为空，也不能重复。

2. 外键约束(FOREIGN KEY)

外键约束体现参照完整性。外键的取值或者为空或者参考父表的主键。

3. 属性约束

属性约束体现了用户定义的完整性。属性约束主要限制某一属性的取值范围。属性约束可分为以下几类：

(1) 非空约束(NOT NULL)——要求某一属性的值不允许为空值。

(2) 唯一约束(UNIQUE)——要求某一属性的值不允许重复。

(3) 检查约束(CHECK)——CHECK 约束可以对某一个属性列的值加以限制。限制就是给某一列设定条件,只有满足条件的值才允许插入。

基本表的完整性约束可定义为两级：表级约束和列级约束。表级约束可以约束表中的任意一列或多列,而列级约束只能约束其所在的某一列。

上述几种约束条件均可作为列级完整性约束条件,但非空约束不可以作为表级完整性约束条件,而其他约束可以作为表级完整性约束条件。

3.2.3　基本表的定义

表是数据库中最基本的操作对象,是实际存放数据的地方。其他的数据库对象的创建及各种操作都是围绕表进行的,可以将表看作含列和行的表单。

SQL 语言使用 CREATE TABLE 语句定义基本表。其一般格式为：

```
CREATE TABLE <基本表名>
        (<列名> <数据类型> [列级完整性约束]
         [,<列名> <数据类型> [列级完整性约束] ]
              …
         [,表级完整性约束]);
```

说明：

(1) 其中,“＜＞”中的内容是必选项,“[ ]”中的内容是可选项。本书以下各章节也遵循这个规定。

(2) ＜基本表名＞——规定了所定义的基本表的名字,在一个用户中不允许有两个基本表的名字相同。＜列名＞——规定了该列(属性)的名称。一个表中不能有两列的名字相同。

(3) 表名或列名命名规则——第一个字符必须是字母,后面可以跟字母、数字、三个特殊符号(_、$、#)；表名或列名中不可以包含空格；表名和列名不区分大小写,但显示出来都是大写；保留字不能用作表名或列名。

(4) ＜数据类型＞——规定了该列的数据类型。

(5) ＜列级完整性约束＞——是指对某一列设置的约束条件。

(6) ＜表级完整性约束＞——规定了关系主键、外键和用户自定义完整性约束。

例题 3.1 创建一个学生表 student，要求所有约束条件均为列级完整性约束，学生表的结构如表 3.3 所示。

表 3.3 student(学生表)

字段名	字段类型	是否为空	说　　明	字段描述
sno	CHAR(8)	NOT NULL	主键	学生学号
sname	VARCHAR2(20)		唯一	学生姓名
sex	CHAR(4)	NOT NULL	非空	性别
age	INT		年龄大于 16 岁	年龄
dept	VARCHAR2(20)			学生所在的系别名称

学生表 student 的创建语句如下：

```
CREATE TABLE student
(
    sno CHAR(8) PRIMARY KEY,              /* 主键约束 */
    sname VARCHAR2(20) UNIQUE,            /* 唯一约束 */
    sex CHAR(4) NOT NULL,                 /* 非空约束 */
    age INT CHECK(Age>16),                /* 检查约束 */
    dept VARCHAR2(20)
);
```

例题 3.2 创建一个课程表 course，要求所有约束条件均为列级完整性约束，课程表的结构如表 3.4 所示。

表 3.4 course(课程表)

字段名	字段类型	是否为空	说　　明	字段描述
cno	CHAR(8)	NOT NULL	主键	课程编号
cname	VARCHAR2(20)	NOT NULL	非空	课程名称
tname	VARCHAR2(20)			授课教师名
cpno	CHAR(8)		外键(参照课程表中的课程编号)	先修课程号
credit	NUMBER			学分

课程表 course 的创建语句如下：

```
CREATE TABLE course
(
    cno CHAR(8) PRIMARY KEY,              /* 主键约束 */
    cname VARCHAR2(20) NOT NULL,          /* 非空约束 */
    tname VARCHAR2(20),
    cpno CHAR(8) REFERENCES course(cno),  /* 外键约束 */
    credit NUMBER
);
```

例题 3.3　创建一个选课表 sc，要求所有约束条件均为表级完整性约束，选课表的结构如表 3.5 所示。

表 3.5　sc(选课表)

字段名	字段类型	是否为空	说　　明	字段描述
sno	CHAR(8)	NOT NULL	外键(参照学生表中的学生编号)	学生学号
cno	CHAR(8)	NOT NULL	外键(参照课程表中的课程编号)	课程编号
grade	NUMBER			选修成绩

其中，(sno，cno)属性组合为主键。

选课表 sc 的创建语句如下：

```
CREATE TABLE sc
(
    sno CHAR(8),
    cno CHAR(8),
    grade NUMBER,
    PRIMARY KEY(sno,cno),                                /* 主键约束 */
    FOREIGN KEY(sno) REFERENCES student(sno),            /* 外键约束 */
    FOREIGN KEY (cno) REFERENCES course(cno)             /* 外键约束 */
);
```

3.2.4　基本表的修改

随着应用环境和实际需求的变化，经常需要修改基本表的结构，包括修改属性列的数据类型及其精度、增加新的属性列或删除属性列、增加新的约束条件或删除原有的约束条件。SQL 语言通过 ALTER TABLE 命令对基本表的结构进行修改。其一般格式为：

```
ALTER TABLE <基本表名>
    [ADD <新列名> <数据类型> [列级完整性约束]]
    [DROP COLUMN <列名>]
    [MODIFY <列名> <新的数据类型>]
    [ADD CONSTRAINT <完整性约束>]
    [DROP CONSTRAINT <完整性约束>];
```

说明：

(1) ADD——为一个基本表增加新的属性列，但新的属性列的值必须允许为空(除非有默认值)。

(2) DROP COLUMN——删除基本表中原有的一列。

(3) MODIFY——修改基本表中原有属性列的数据类型。

(4) ADD CONSTRAINT 和 DROP CONSTRAINT 分别表示添加完整性约束和删除完整性约束。

(5) 以上的命令格式在实际的数据库管理系统中可能有所不同,用户在使用时应参阅实际数据库系统的参考手册。

例题 3.4 向 student 表中增加一个身高属性列 height,数据类型为 INT。

```
ALTER TABLE student ADD height INT;
```

新增加的属性列总是表的最后一列。不论表中是否已经有数据,新增加的列值为空。所以新增加的属性列不能有 NOT NULL 约束,否则就会产生矛盾。

例题 3.5 将 student 表中的 height 属性列的数据类型改为 REAL。

```
ALTER TABLE student MODIFY height REAL;
```

修改原有的列定义有可能会破坏已有数据,所以在修改时需要注意:可以增加列值的宽度及小数点的长度,只有当某列所有行的值为空或整张表是空时,才能减少其列值宽度,或改变其列值的数据类型。

例题 3.6 给 student 表中 height 属性列增加一个检查约束,约束的名字为 CHK_HEIGHT,要求学生的身高要超过 140cm 才行。

```
ALTER TABLE student ADD CONSTRAINT CHK_HEIGHT CHECK(height > 140);
```

例题 3.7 删除 height 属性列上的 CHECK 约束。

```
ALTER TABLE student DROP CONSTRAINT CHK_HEIGHT;
```

例题 3.8 删除 student 表中新增加的 height 属性列。

```
ALTER TABLE student DROP COLUMN height;
```

3.2.5 基本表的删除

当数据库某个基本表不再使用时,可以使用 DROP TABLE 语句删除它。其一般格式为:

```
DROP TABLE <表名> [CASCADE CONSTRAINTS];
```

删除基本表时要注意以下几点:

(1) 表一旦被删除,则无法恢复。

(2) 如果表中有数据,则表的结构连同数据一起删除。

(3) 在表上的索引、约束条件、触发器,以及表上的权限也一起被删除。

(4) 当删除表时,涉及该表的视图、存储过程、函数、包被设置为无效。

(5) 只有表的创建者或者拥有 DROP ANY TABLE 权限的用户才能删除表。

(6) 如果两张表之间有主外键约束条件,则必须先删除子表,然后再删除主表。

(7) 如果加上 CASCADE CONSTRAINTS,在删除基本表的同时,相关的依赖对象也

一起被删除。

例题 3.9　删除学生选课表 sc。

```
DROP TABLE sc;
```

基本表定义一旦被删除,表中的数据、此表上建立的索引和视图都将自动被删除掉。因此执行删除基本表的操作一定要格外小心。但在有的系统(如 Oracle)中,删除基本表后建立在此表上的视图定义仍然保留在数据字典中,但是,当用户引用时就会报错。

3.2.6　索引的定义和删除

基本表建立并存放数据后,如果数据相当多,则 DBMS 会在顺序扫描上耗费很长的时间,这样将大大影响查询效率。为了解决查询速度问题,需要在针对数据表的查询字段上定义索引。索引提供了一种直接、快速访问记录的方式,可以大大提高数据查询速度。

索引是根据表中一列或若干列按照一定顺序建立的列值与记录行之间的对应关系表。索引属于物理存储的路径概念,而不是用户使用的逻辑概念。建立在多个列上的索引被称为复合索引。系统在存取数据时会自动选择合适的索引作为存取路径,用户不必也不能显式地选择索引。

有两种重要的索引:聚簇索引(clustered index)和非聚簇索引(non-clustered index)。

聚簇索引确定表中数据的物理顺序。聚簇索引类似于按姓氏排列数据的电话簿。由于聚簇索引规定数据在表中的物理存储顺序,因此一个表只能包含一个聚簇索引。建立聚簇索引后,更新索引列数据时,往往导致表中记录的物理顺序的变更,代价较大,因此对于经常更新的列不宜建立聚簇索引。

非聚簇索引是数据存储在一个地方,索引存储在另一个地方,索引带有指针指向数据的存储位置。索引中的项目按索引键值的顺序存储,而表中的信息按另一种顺序存储。

1. 索引的定义

在 SQL 语言中,建立索引使用 CREATE INDEX 语句,其一般格式为:

```
CREATE [UNIQUE] [CLUSTER] INDEX <索引名>
ON <基本表名> (<列名> [<次序>],[,<列名> [<次序>]] …);
```

说明:

(1) UNIQUE:规定此索引为唯一性索引。每一个索引值只对应于表中唯一的记录。

(2) CLUSTER:规定此索引为聚簇索引。省略 CLUSTER 则表示创建的索引为非聚簇索引。

(3) <次序>:建立索引时指定列名的索引表是 ASC(升序)或 DESC(降序)。若不指定,默认为升序。

例题 3.10　为 student、course、sc 三张表建立索引。其中 student 表按学号 sno 升序建唯一索引, course 表按课程号 cno 降序建唯一索引,sc 表按学号 sno 升序和课程号 cno 降

序建唯一索引。

```
CREATE UNIQUE INDEX index_stu ON student(sno ASC);
CREATE UNIQUE INDEX index_cou ON course(cno DESC);
CREATE UNIQUE INDEX index_sc ON sc(sno ASC,cno DESC);
```

2. 索引的删除

索引可以加快查询速度,但如果数据的增、删、改操作很频繁,系统就会花许多时间来维护索引,导致系统开销增加。

索引一经建立,就由系统使用和维护它,不需用户干预。建立索引是为了减少查询操作的时间,但如果数据增加删改频繁,系统会花费许多时间来维护索引。过多的索引甚至会导致索引碎片,降低系统效率。因此不必要的索引应该及时删除。

删除索引的格式为:

```
DROP INDEX <索引名>;
```

例题 3.11 删除 course 表的 index_cou 索引。

```
DROP INDEX index_cou;
```

删除索引时,系统会同时从数据字典中删除有关该索引的描述。

3.3 数据查询

3.3.1 SELECT 语句格式

SQL 语言中最重要、最核心的操作就是数据查询。关系代数的运算在关系数据库中主要由 SQL 数据查询来体现。SQL 语言提供 SELECT 语句进行数据库的查询,该语句具有灵活的使用方式和丰富的功能。其基本格式为:

```
SELECT [ALL|DISTINCT] <目标列表达式> [,<目标列表达式>]…
FROM <表名或视图名> [,<表名或视图名>]…
[WHERE <条件表达式>]
[GROUP BY <列名 1> [HAVING <组条件表达式>]]
[ORDER BY <列名 2> [ASC|DESC]];
```

其中:

(1) SELECT 子句说明要查询的数据。ALL 表示筛选出数据库表中满足条件的所有记录,一般情况下省略不写。DISTINCT 表示输出结果中无重复记录。

(2) FROM 子句说明要查询的数据来源。可以是数据库中的一个或多个表或视图,各项之间用逗号分隔。

(3) WHERE 子句指定查询条件。查询条件中会涉及 SQL 函数和 SQL 操作符。

(4) GROUP BY 子句表示在查询时,可以按照某个或某些字段分组汇总,各分组选项之间用逗号分隔。HAVING 子句必须跟随 GROUP BY 一起使用,表示在分组汇总时,可以根据组条件表达式筛选出满足条件的组记录。

(5) ORDER BY 子句表示在显示结果时,按照指定字段进行排序。ASC 表示升序,DESC 表示降序,省略不写默认情况下是 ASC。

整个 SELECT 语句的含义是:根据 WHERE 子句的条件表达式,从 FROM 子句指定的表或视图中找出满足条件的元组,再按照 SELECT 子句中的目标列表达式,选出元组中的属性值形成结果表。如果有 GROUP BY 子句,则将结果按<列名 1>的值进行分组,该属性列值相等的元组为一个组。通常会在每组中使用聚集函数。如果 GROUP BY 子句带有 HAVING 子句,则只有满足指定条件的组才能够输出。如果有 ORDER BY 子句,则结果表还需要按<列名 2>的值的升序或者降序排列。查询子句的顺序是不可以前后调换的。

由于 SELECT 语句的形式多样,可以完成单表查询、多表连接查询、嵌套查询和集合查询等,想要熟练地掌握和运用 SELECT 语句必须要下一番工夫。下面将通过大量的例子来介绍 SELECT 语句的功能。

下面以学生选课系统为例说明 SELECT 语句的各种用法。学生选课系统中包含三张表。

(1) 学生表:student(sno,sname,sex,age,dept),表中属性列依次是学生学号、姓名、性别、年龄、学生所在的系别名称。其中 sno 为主键,表中数据如图 3.1 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 3.1　学生基本信息

(2) 课程表:course(cno,cname,tname,cpno,credit),表中属性列依次是课程编号、课程名称、授课教师名、先修课程号和学分。其中 cno 为主键,表中数据如图 3.2 所示。

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图 3.2　课程基本信息

(3) 选课表：sc(sno,cno,grade),表中属性列依次是学号、课程号和成绩。其中属性组合(sno,cno)为主键,表中数据如图3.3所示。

SNO	CNO	GRADE
11432001	c1	75
11432001	c2	95
11432002	c1	82
11432002	c2	88
11432002	c4	76
11432002	c5	55
11432003	c1	65
11432003	c2	72
11432003	c3	90
11432003	c4	85
11432004	c1	93
11432004	c2	96
11432004	c5	91
11432005	c1	50
11432005	c2	70
11432005	c5	88
11432005	c8	43
11432007	c1	75
11432007	c2	73
11432007	c3	66
11432007	c6	82
11432007	c7	94
11432008	c1	82
11432008	c2	77
11432008	c3	85
11432008	c4	87
11432008	c5	82
11432008	c6	94
11432008	c7	92
11432008	c8	89
11432009	c1	86
11432009	c2	
11432009	c8	
11432010	c1	73
11432010	c2	
11432010	c8	76

图3.3 选课基本信息

3.3.2 单表无条件查询

单表查询是指查询的数据只来自一张表,此时,SELECT语句中的FROM子句只涉及一张表的查询。

1. 选择表中若干列

选择表中的全部列或部分列,这就是投影运算。

1) 查询指定的列

例题3.12 查询全体学生的学号、姓名和年龄。

```
SELECT sno,sname,age
FROM student;
```

结果如下：

SNO	SNAME	AGE
11432001	陈一	17
11432002	姚二	20
11432003	张三	19
11432004	李四	22
11432005	王五	22
11432006	赵六	19
11432007	陈七	23
11432008	刘八	21
11432009	张九	18
11432010	孙十	21

例题 3.13　查询全部课程的课程名称和授课教师名。

```
SELECT cname,tname
FROM course;
```

结果如下：

CNAME	TNAME
大学英语	李老师
马克思主义基本原理	赵老师
高等数学	曹老师
证据学	杨老师
罪犯心理矫治	张老师
JAVA语言程序设计	唐老师
JSP程序设计	唐老师
公共安全危机管理	刘老师

2）查询全部列

例题 3.14　查询全部课程的详细记录。

```
SELECT *
FROM course;
```

结果如下：

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

当所查询的列是关系的所有属性时，可以使用 * 来表示所显示的列。

3）查询经过计算的值

例题 3.15　查询全体学生的姓名、性别及其出生年份。

```
SELECT sname,sex,2015 - age
FROM student;
```

结果如下：

```
SNAME                SEX  2015-AGE
-------------------- ---- ----------
陈一                 男         1998
姚二                 女         1995
张三                 女         1996
李四                 男         1993
王五                 男         1993
赵六                 男         1996
陈七                 女         1992
刘八                 男         1994
张九                 女         1997
孙十                 女         1994
```

4）指定别名来改变查询结果的列标题

从前面的查询结果中可以看到，显示的每一个属性列的标题就是列名，有时候列名就是拼音代码，含义不是很清楚，为了解决这个问题，我们可以给属性列提供一个别名。

方法就是：在列名的后面加上一个空格或者是“as”，然后写上它的别名。在查询结果显示时就用别名代替列名了。

例题 3.16 查询全体学生的姓名、性别及其出生年份。

```
SELECT sname,sex,2015 - age 出生年份
FROM student;
```

结果如下：

```
SNAME                SEX    出生年份
-------------------- ---- ----------
陈一                 男         1998
姚二                 女         1995
张三                 女         1996
李四                 男         1993
王五                 男         1993
赵六                 男         1996
陈七                 女         1992
刘八                 男         1994
张九                 女         1997
孙十                 女         1994
```

2. 选择表中若干行

选择表中若干行，这就是选择运算。这里介绍无条件的选择运算，后面介绍有条件的选择运算，需要注意的是，应消除取值重复的行。

例题 3.17 查询所有选修了课程的学生学号。

```
SELECT sno
FROM sc;
```

结果如下：

```
SNO
--------
11432001
11432001
11432002
11432002
11432002
11432002
11432003
11432003
11432003
11432003
11432004
11432004
11432004
11432005
11432005
11432005
11432005
11432007
11432007
11432007
11432007
11432007
11432008
11432008
11432008
11432008
11432008
11432008
11432008
11432008
11432009
11432009
11432009
11432010
11432010
11432010
```

由于存在一名同学选修多门课程的情况，所以查询的结果中包含了许多重复的行。如果想去掉重复的行，必须指定 DISTINCT 关键字。

```
SELECT DISTINCT sno
FROM sc;
```

结果如下：

```
SNO
--------
11432001
11432002
11432003
11432004
11432005
11432007
11432008
11432009
11432010
```

3.3.3　单表有条件查询

单表查询时，若需满足某些条件则可以通过 WHERE 子句来实现。使用 WHERE 子句时，应该注意以下几点：

（1）如果该列数据类型为字符型，需要使用单引号把字符串括起来。如 WHERE cname＝'JSP 程序设计'，单引号内的字符串大小写是有区别的。

（2）如果该列数据类型为日期型，需要使用单引号把日期括起来。

（3）如果该列数据类型为数值型，则不必用单引号。如 WHERE age＞20。

（4）在 WHERE 子句中可以使用列名或表达式，但不能使用它的别名。

WHERE 子句常用的查询条件如表 3.6 所示。

表 3.6　常用的查询条件

查询条件	谓　词
比较	＝，＞，＜，＞＝，＜＝，！＝，＜＞，！＞，！＜，NOT 等比较运算符
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空值	IS NULL，IS NOT NULL
多重条件	AND，OR

1. 比较大小

例题 3.18　查询公安信息系全体学生的姓名。

```
SELECT sname
FROM student
WHERE dept = '公安信息系';
```

结果如下：

```
SNAME
------
陈七
刘八
```

例题 3.19　查询年龄超过 20 岁的学生姓名及其年龄。

```
SELECT sname,age
FROM student
WHERE age > 20;
```

结果如下：

```
SNAME                       AGE
-------------------- ----------
李四                         22
王五                         22
陈七                         23
刘八                         21
孙十                         21
```

例题 3.20　查询考试成绩有不及格的学生的学号。

```
SELECT DISTINCT sno
FROM sc
WHERE grade < 60;
```

结果如下：

```
SNO
--------
11432005
11432002
```

语句中使用了 DISTINCT 关键字，目的是当某一个学生有多门课程不及格时，他的学号只显示一次。

2. 确定范围(谓词 BETWEEN AND)

例题 3.21　查询年龄在 16～20 岁(包括 16 岁和 20 岁)之间的学生姓名和年龄。

```
SELECT sname,age
FROM student
WHERE age BETWEEN 16 AND 20;
```

结果如下：

```
SNAME                       AGE
-------------------- ----------
陈一                         17
姚二                         20
张三                         19
赵六                         19
张九                         18
```

例题 3.22　查询年龄不在 16～20 岁之间的学生姓名和年龄。

```
SELECT sname,age
FROM student
WHERE age NOT BETWEEN 16 AND 20;
```

结果如下：

```
SNAME                       AGE
-------------------- ----------
李四                         22
王五                         22
陈七                         23
刘八                         21
孙十                         21
```

3. 确定集合(谓词 IN)

例题 3.23　查询侦查系、公安信息系和治安管理系的学生姓名和性别。

```
SELECT sname,sex
FROM student
WHERE dept IN ('侦查系','公安信息系','治安管理系');
```

结果如下：

```
SNAME                SEX
-------------------- ---
陈一                 男
姚二                 女
张三                 女
陈七                 女
刘八                 男
张九                 女
孙十                 女
```

例题 3.24 查询既不是侦查系、公安信息系,也不是治安管理系的学生姓名和性别。

```
SELECT sname,sex
FROM student
WHERE dept NOT IN ('侦查系','公安信息系','治安管理系');
```

结果如下:

SNAME	SEX
李四	男
王五	男
赵六	男

4. 字符匹配(谓词 LIKE)

谓词 LIKE 可以用来进行字符串的匹配。基本格式为:

```
[NOT] LIKE '<匹配串>' [ESCAPE '<换码字符>']
```

其含义是查找指定的属性列值与<匹配串>相匹配的元组。<匹配串>可以是一个完整的字符串,也可以含有通配符%和_。

其中,%(百分号)代表任意长度(长度可以为 0)的字符串;_(下横线)代表任意单个字符。

例题 3.25 查询所有姓张的学生姓名、年龄和系别名称。

```
SELECT sname,age,dept
FROM student
WHERE sname LIKE '张%';
```

结果如下:

SNAME	AGE	DEPT
张三	19	侦查系
张九	18	治安管理系

如果换成 NOT LIKE,表示不姓张的同学。

例题 3.26 查询以"JSP"开头且倒数第 2 个字符为"设"的课程详细信息。

```
SELECT *
FROM course
WHERE cname LIKE 'JSP%设_';
```

结果如下:

CNO	CNAME	TNAME	CPNO	CREDIT
c7	JSP程序设计	唐老师	c6	2

如果用户查询的匹配字符串本身就含有%或_,这时就要使用 ESCAPE'<换码字符>'短语对通配符进行转义。上例中,如果要查询以"JSP_"开头且倒数第 2 个字符为"设"的课

程详细信息，则需要将条件改写成

```
WHERE cname LIKE 'JSP\_%设_' ESCAPE'\';
```

5. 涉及空值的查询

例题 3.27　查询选修了课程但没有成绩的学生学号和相应的课程号。

```
SELECT sno,cno
FROM sc
WHERE grade IS NULL;
```

结果如下：

```
SNO      CNO
-------- ----
11432009 c2
11432009 c8
11432010 c2
```

注意：这里“IS”不能用等号（=）代替。

例题 3.28　查询选修了课程并且有成绩的学生学号和相应的课程号。

```
SELECT sno,cno
FROM sc
WHERE grade IS NOT NULL;
```

结果如下：

```
SNO      CNO
-------- ----
11432001 c1
11432001 c2
11432002 c1
11432002 c2
11432002 c4
11432002 c5
11432003 c1
11432003 c2
11432003 c3
11432003 c4
11432004 c1
11432004 c2
11432004 c5
11432005 c1
11432005 c2
11432005 c5
11432005 c8
11432007 c1
11432007 c2
11432007 c3
11432007 c6
11432007 c7
11432008 c1
11432008 c2
```

```
11432008 c3
11432008 c4
11432008 c5
11432008 c6
11432008 c7
11432008 c8
11432009 c1
11432010 c1
11432010 c8
```

6. 多重条件查询

逻辑运算符 AND 和 OR 可用来连接多个查询条件。AND 的优先级高于 OR，但用户可以通过括号来改变优先级。

例题 3.29 查询侦查系女同学的姓名和年龄。

```
SELECT sname,age
FROM student
WHERE dept = '侦查系' AND sex = '女';
```

结果如下：

```
SNAME                       AGE
-------------------- ----------
姚二                         20
张三                         19
```

例题 3.30 查询治安管理系或年龄在 20 岁以下的学生姓名。

```
SELECT sname
FROM student
WHERE dept = '治安管理系' OR age < 20;
```

结果如下：

```
SNAME
------
陈一
张三
赵六
张九
孙十
```

3.3.4 聚集函数

为了进一步方便用户，增强检索功能，SQL 提供了许多聚集函数，主要有：

(1) COUNT ([DISTINCT|ALL] *) 统计元组个数

COUNT ([DISTINCT|ALL] <列名>) 统计某一列中值的个数

(2) SUM ([DISTINCT|ALL] <列名>) 计算一列值的总和(此列必须是数值型)

(3) AVG ([DISTINCT|ALL] <列名>) 计算一列值的平均值(此列必须是数值型)

(4) MAX ([DISTINCT|ALL] <列名>) 求一列值中的最大值

(5) MIN ([DISTINCT|ALL] ＜列名＞)　求一列值中的最小值

如果指定 DISTINCT 短语，则表示在查询时要取消指定列中的重复值。如果不指定 DISTINCT 短语或指定 ALL 短语(ALL 为默认值)，则表示不取消重复值。

在聚集函数遇到空值时，除 COUNT(*)外，都会跳过空值而只处理非空值。

例题 3.31　查询学生表中的总人数。

```
SELECT COUNT( * )
FROM student;
```

结果如下：

```
COUNT(*)
----------
        10
```

例题 3.32　查询选修了课程的学生人数。

```
SELECT COUNT(DISTINCT sno)
FROM sc;
```

结果如下：

```
COUNT(DISTINCTSNO)
------------------
                 9
```

由于存在一个同学选修多门课程的情况，为了避免重复计算学生人数，所以必须加 DISTINCT 关键字，表示在统计人数时，取消指定列中的重复值。

例题 3.33　查询选修 c3 课程的平均成绩、最高成绩和最低成绩。

```
SELECT AVG(grade),MAX(grade),MIN(grade)
FROM sc
WHERE cno = 'c3';
```

结果如下：

```
AVG(GRADE) MAX(GRADE) MIN(GRADE)
---------- ---------- ----------
80.3333333         90         66
```

例题 3.34　查询学号为 114032001 的学生选修课程的成绩总和。

```
SELECT SUM(grade)
FROM sc
WHERE sno = '11432001';
```

结果如下：

```
SUM(GRADE)
----------
       170
```

3.3.5 分组查询和排序查询

1. 对查询结果分组

在 SELECT 语句中可以使用 GROUP BY 子句将查询结果按照某一列或多列的值分组，值相等的为一组，然后使用聚集函数返回每一个组的汇总信息。而且可以使用 HAVING 子句限制返回的结果集。

例题 3.35 查询选课表中每门课程的课程号及这门课程的选修人数。

```
SELECT cno, COUNT(sno)
FROM sc
GROUP BY cno;
```

结果如下：

```
CNO      COUNT(SNO)
-------- ----------
c8                4
c1                9
c3                3
c7                2
c4                3
c6                2
c5                4
c2                9
```

该 SELECT 语句对 sc 表按 cno 的值进行分组，所有相同 cno 值的元组为一组，然后对每一组用聚集函数 COUNT 来计算，统计该组的学生人数。

在分组查询中 HAVING 子句用于分完组后，对每一组进行条件判断，只有满足条件的分组才被选出来，这种条件判断一般与 GROUP BY 子句有关。

例题 3.36 查询选修 5 门及其以上课程的学生学号。

```
SELECT sno
FROM sc
GROUP BY sno
HAVING COUNT(cno)>=5;
```

结果如下：

```
SNO
--------
11432007
11432008
```

使用 GROUP BY 和 HAVING 子句时需要注意以下几点：

(1) 带有 GROUP BY 子句的查询语句中，在 SELECT 子句中指定的列要么是 GROUP BY 子句中指定的列，要么包含聚集函数，否则出错。

(2) 可以使用多个属性列进行分组。

(3) 聚集函数只能够出现在 SELECT、HAVING、ORDER BY 子句中。在 WHERE 子

句中是不能使用聚集函数的。

在一个 SELECT 语句中可以有 WHERE 子句和 HAVING 子句,这两个子句都可以用于限制查询的结果。WHERE 子句与 HAVING 子句的区别是:

(1) WHERE 子句的作用是在分组之前过滤数据。WHERE 条件中不能包含聚集函数。使用 WHERE 条件选择满足条件的行。

(2) HAVING 子句的作用是在分组之后过滤数据。HAVING 条件中经常包含聚集函数。使用 HAVING 条件选择满足条件的组。使用 HAVING 子句时必须首先使用 GROUP BY 进行分组。

2. 对查询结果进行排序

ORDER BY 子句可指定按照一个或多个属性列的升序(ASC)或者降序(DESC)重新排列查询结果。省略不写,默认为升序排列。由于是控制输出结果,因此 ORDER BY 子句只能用于最终的查询结果。

例题 3.37　查询选修 c3 课程的学生学号及成绩,查询结果按照成绩的降序排列。

```
SELECT sno, grade
FROM sc
WHERE cno = 'c3'
ORDER BY grade DESC;
```

结果如下:

SNO	GRADE
11432003	90
11432008	85
11432007	66

例题 3.38　查询所有学生的基本信息,查询结果按学生年龄的降序排列,年龄相同时则按学号升序排列。

```
SELECT *
FROM  student
ORDER BY  age DESC, sno ASC;
```

结果如下:

SNO	SNAME	SEX	AGE	DEPT
11432007	陈七	女	23	公安信息系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432008	刘八	男	21	公安信息系
11432010	孙十	女	21	治安管理系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432006	赵六	男	19	刑事技术系
11432009	张九	女	18	治安管理系
11432001	陈一	男	17	侦查系

3.3.6 连接查询

在数据库中通常存在着多个相互关联的表，用户常常需要同时从多个表中找出自己想要的数据，这就涉及多个数据表的查询。

连接查询是指通过两个或两个以上的关系表或视图的连接操作来实现的查询。连接查询是关系数据库中最主要的查询，包括等值连接、非等值连接、自然连接、自身连接、外连接和复合条件连接等。

连接查询中用来连接两个表的条件称为连接条件或连接谓词，其格式为：

```
[<表名 1>.]<列名 1> <比较运算符> [<表名 2>.]<列名 2>
```

其中比较运算符主要有=、>、<、>=、<=、!=。

此外连接谓词还可以使用下面形式：

```
[<表名 1>.]<列名 1> BETWEEN [<表名 2>.]<列名 2> AND [<表名 2>.]<列名 3>
```

连接条件中的列名称为连接字段。连接条件中的各连接字段类型必须是可比的，但名字不必相同。

1. 等值连接

当连接运算符为“=”时，称为等值连接。使用其他运算符时，称为非等值连接。

例题 3.39 查询每个同学基本信息及其选修课程的情况。

```
SELECT student.*, sc.*
FROM student, sc
WHERE student.sno = sc.sno;
```

结果如下：

SNO	SNAME	SEX	AGE	DEPT	SNO	CNO	GRADE
11432001	陈一	男	17	侦查系	11432001	c1	75
11432001	陈一	男	17	侦查系	11432001	c2	95
11432002	姚二	女	20	侦查系	11432002	c1	82
11432002	姚二	女	20	侦查系	11432002	c2	88
11432002	姚二	女	20	侦查系	11432002	c4	76
11432002	姚二	女	20	侦查系	11432002	c5	55
11432003	张三	女	19	侦查系	11432003	c1	65
11432003	张三	女	19	侦查系	11432003	c2	72
11432003	张三	女	19	侦查系	11432003	c3	90
11432003	张三	女	19	侦查系	11432003	c4	85
11432004	李四	男	22	刑事技术系	11432004	c1	93
11432004	李四	男	22	刑事技术系	11432004	c2	96
11432004	李四	男	22	刑事技术系	11432004	c5	91
11432005	王五	男	22	刑事技术系	11432005	c1	50
11432005	王五	男	22	刑事技术系	11432005	c2	70
11432005	王五	男	22	刑事技术系	11432005	c5	88
11432005	王五	男	22	刑事技术系	11432005	c8	43
11432007	陈七	女	23	公安信息系	11432007	c1	75
11432007	陈七	女	23	公安信息系	11432007	c2	73
11432007	陈七	女	23	公安信息系	11432007	c3	66
11432007	陈七	女	23	公安信息系	11432007	c6	82
11432007	陈七	女	23	公安信息系	11432007	c7	94
11432008	刘八	男	21	公安信息系	11432008	c1	82
11432008	刘八	男	21	公安信息系	11432008	c2	77

11432008	刘八	男	21	公安信息系	11432008	c3	85
11432008	刘八	男	21	公安信息系	11432008	c4	87
11432008	刘八	男	21	公安信息系	11432008	c5	82
11432008	刘八	男	21	公安信息系	11432008	c6	94
11432008	刘八	男	21	公安信息系	11432008	c7	92
11432008	刘八	男	21	公安信息系	11432008	c8	89
11432009	张九	女	18	治安管理系	11432009	c1	86
11432009	张九	女	18	治安管理系	11432009	c2	
11432009	张九	女	18	治安管理系	11432009	c8	
11432010	孙十	女	21	治安管理系	11432010	c1	73
11432010	孙十	女	21	治安管理系	11432010	c2	
11432010	孙十	女	21	治安管理系	11432010	c8	76

说明：

（1）student.sno＝sc.sno 是两个关系表的连接条件，student 表和 sc 表中的记录只有满足这个条件才能连接。

（2）在 student 表和 sc 表中存在相同的属性名 sno，因此存在属性的二义性问题。SQL 通过在属性前面加上关系名及一个小圆点来解决这个问题，表示该属性来自这个关系。

2. 自然连接

如果是按照两个表中的相同属性进行等值连接，并且在结果中去掉了重复的属性列，我们称之为自然连接。

例题 3.40　用自然连接来完成查询每个同学基本信息及其选修课程的情况。

```
SELECT  student.sno,sname,sex,age,dept,cno,grade
FROM  student,sc
WHERE  student.sno = sc.sno;
```

结果如下：

SNO	SNAME	SEX	AGE	DEPT	CNO	GRADE
11432001	陈一	男	17	侦查系	c1	75
11432001	陈一	男	17	侦查系	c2	95
11432002	姚二	女	20	侦查系	c1	82
11432002	姚二	女	20	侦查系	c2	88
11432002	姚二	女	20	侦查系	c4	76
11432002	姚二	女	20	侦查系	c5	55
11432003	张三	女	19	侦查系	c1	65
11432003	张三	女	19	侦查系	c2	72
11432003	张三	女	19	侦查系	c3	90
11432003	张三	女	19	侦查系	c4	85
11432004	李四	男	22	刑事技术系	c1	93
11432004	李四	男	22	刑事技术系	c2	96
11432004	李四	男	22	刑事技术系	c5	91
11432005	王五	男	22	刑事技术系	c1	50
11432005	王五	男	22	刑事技术系	c2	70
11432005	王五	男	22	刑事技术系	c5	88
11432005	王五	男	22	刑事技术系	c8	43
11432007	陈七	女	23	公安信息系	c1	75
11432007	陈七	女	23	公安信息系	c2	73
11432007	陈七	女	23	公安信息系	c3	66
11432007	陈七	女	23	公安信息系	c6	82
11432007	陈七	女	23	公安信息系	c7	94
11432008	刘八	男	21	公安信息系	c1	82
11432008	刘八	男	21	公安信息系	c2	77
11432008	刘八	男	21	公安信息系	c3	85
11432008	刘八	男	21	公安信息系	c4	87
11432008	刘八	男	21	公安信息系	c5	82

```
11432008 刘八    男    21 公安信息系    c6    94
11432008 刘八    男    21 公安信息系    c7    92
11432008 刘八    男    21 公安信息系    c8    89
11432009 张九    女    18 治安管理系    c1    86
11432009 张九    女    18 治安管理系    c2
11432009 张九    女    18 治安管理系    c8
11432010 孙十    女    21 治安管理系    c1    73
11432010 孙十    女    21 治安管理系    c2
11432010 孙十    女    21 治安管理系    c8    76
```

3. 复合条件连接

上面例题中，在 WHERE 子句里除了连接条件外，还可以有多个限制条件。连接条件用于多个表之间的连接，限制条件用于限制所选取的记录要满足什么条件，这种连接称为复合条件连接。

例题 3.41 查询选修课程号为 c1，并且成绩不及格的学生学号、姓名和系别名称。

```
SELECT student.sno, sname, dept
FROM student, sc
WHERE student.sno = sc.sno                /* 连接条件 */
      AND cno = 'c1'                      /* 限制条件 */
      AND grade < 60;                     /* 限制条件 */
```

结果如下：

```
SNO       SNAME                DEPT
--------- -------------------- ----------
11432005  王五                 刑事技术系
```

连接操作除了可以是两个表的连接外，还可以是两个以上的表进行连接，把它称为多表连接。

例题 3.42 查询侦查系选修“高等数学”课程的学生姓名、授课教师名以及这门课程的成绩。

```
SELECT sname, tname, grade
FROM student, course, sc
WHERE student.sno = sc.sno                /* 连接条件 */
      AND course.cno = sc.cno             /* 连接条件 */
      AND dept = '侦查系'                 /* 限制条件 */
      AND cname = '高等数学';             /* 限制条件 */
```

结果如下：

```
SNAME                TNAME                     GRADE
-------------------- -------------------- ----------
张三                 曹老师                       90
```

如果是多个表之间连接，那么 WHERE 子句中就有多个连接条件。n 个表之间的连接至少有 $n-1$ 个连接条件。

4. 自身连接

连接操作不仅可以在两个表之间进行，也可以是一个表与其自身进行连接，称为表的自身连接。自身连接要求必须给表取别名，把它们当作两个不同的表来处理。

例题 3.43　在 sc 表中查询至少选修课程号为 c3 和 c4 的学生学号。

在 sc 表中，每一条记录只显示一个学生选修一门课程的情况，在这里，一条记录是不能够同时显示选修两门课程的情况，因此就要将 sc 表与其自身连接。为 sc 表取两个别名：一个是 x，另一个是 y。完成查询的语句为：

```
SELECT x.sno
FROM sc x, sc y
WHERE x.sno = y.sno                           /* 连接条件 */
      AND x.cno = 'c3'                        /* 限制条件 */
      AND y.cno = 'c4';                       /* 限制条件 */
```

结果如下：

```
SNO
--------
11432003
11432008
```

在例题 3.43 中，连接条件用来实现每一条记录是同一个学生的选课信息，限制条件用来实现选修的课程至少有 c3 和 c4。

5. 外连接

在通常的连接操作中，只有满足连接条件的元组才能作为结果输出。如例题 3.39 和例题 3.40 的结果中没有 11432006 赵六学生的信息，原因在于他没有选课，在 sc 表中没有相应的元组。如果想以 student 表为主体列出每个学生的基本情况及其选课情况，若某个学生没有选课，只输出学生的基本信息，其选课信息可以为空值，此时就需要使用外连接了。

外连接的表示方法为，在连接条件的某一边加上操作符(＋)(有的数据库系统中用 *)。(＋)号放在连接条件中信息不完整的那一边。外连接运算符(＋)出现在连接条件的右边，则称为左外连接；若出现在连接条件的左边，则称为右外连接。

例题 3.44　以 student 表为主体列出每个学生的基本情况及其选课情况，若某个学生没有选课，只输出学生的基本信息，其选课信息为空值。

```
SELECT student.sno, sname, sex, age, dept, cno, grade
FROM student, sc
WHERE student.sno = sc.sno(+);
```

结果如下：

SNO	SNAME	SEX	AGE	DEPT	CNO	GRADE
11432001	陈一	男	17	侦查系	c1	75
11432001	陈一	男	17	侦查系	c2	95
11432002	姚二	女	20	侦查系	c1	82
11432002	姚二	女	20	侦查系	c2	88
11432002	姚二	女	20	侦查系	c4	76
11432002	姚二	女	20	侦查系	c5	55
11432003	张三	女	19	侦查系	c1	65
11432003	张三	女	19	侦查系	c2	72
11432003	张三	女	19	侦查系	c3	90
11432003	张三	女	19	侦查系	c4	85
11432004	李四	男	22	刑事技术系	c1	93
11432004	李四	男	22	刑事技术系	c2	96
11432004	李四	男	22	刑事技术系	c5	91
11432005	王五	男	22	刑事技术系	c1	50
11432005	王五	男	22	刑事技术系	c2	70
11432005	王五	男	22	刑事技术系	c5	88
11432005	王五	男	22	刑事技术系	c8	43
11432007	陈七	女	23	公安信息系	c1	75
11432007	陈七	女	23	公安信息系	c2	73
11432007	陈七	女	23	公安信息系	c3	66
11432007	陈七	女	23	公安信息系	c6	82
11432007	陈七	女	23	公安信息系	c7	94
11432008	刘八	男	21	公安信息系	c1	82
11432008	刘八	男	21	公安信息系	c2	77
11432008	刘八	男	21	公安信息系	c3	85
11432008	刘八	男	21	公安信息系	c4	87
11432008	刘八	男	21	公安信息系	c5	82
11432008	刘八	男	21	公安信息系	c6	94
11432008	刘八	男	21	公安信息系	c7	92
11432008	刘八	男	21	公安信息系	c8	89
11432009	张九	女	18	治安管理系	c1	86
11432009	张九	女	18	治安管理系	c2	
11432009	张九	女	18	治安管理系	c8	
11432010	孙十	女	21	治安管理系	c1	73
11432010	孙十	女	21	治安管理系	c2	
11432010	孙十	女	21	治安管理系	c8	76
11432006	赵六	男	19	刑事技术系		

3.3.7 嵌套查询

在 SQL 语言中，一个 SELECT-FROM-WHERE 语句称为一个查询块。将一个查询块嵌套在另一个查询块的 WHERE 子句或 HAVING 子句的条件中的查询称为嵌套查询。这也是涉及多表的查询，其中外层查询称为父查询，内层查询称为子查询。

子查询中还可以嵌套其他子查询，即允许多层嵌套查询，其执行过程是由内向外的，每一个子查询是在上一级查询处理之前完成的。这样上一级的查询就可以利用已完成的子查询的结果，可以将一系列简单的查询组合成复杂的查询，从而一些原来无法实现的查询也因为有了多层嵌套的子查询而迎刃而解。

使用子查询的原则：

(1) 子查询必须用括号括起来。

(2) 子查询不能包含 ORDER BY 子句。

(3) 子查询可以在许多 SQL 语句中使用，如 SELECT、INSERT、UPDATE、DELETE 语句中。

1. 不相关子查询

查询条件不依赖于父查询的子查询称为不相关子查询。它的执行过程为：先执行子查询，将子查询的结果作为外层父查询的条件，然后执行父查询。

不相关子查询的特点：

(1) 先执行子查询，后执行父查询。

(2) 子查询能够独立执行，不依赖于外层父查询。

(3) 子查询只执行一次。

1) 带有 IN 谓词的子查询

当子查询的结果是一个集合时，经常使用带 IN 谓词的子查询。

例题 3.45　查询选修课程号为 c4 的学生姓名。

方法一：采用前面学习的多表连接查询来完成。

```
SELECT sname
FROM student,sc
WHERE student.sno = sc.sno AND cno = 'c4';
```

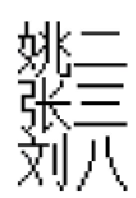
结果如下：

```
SNAME
------
姚二
张三
刘八
```

方法二：采用子查询来完成。

```
SELECT sname
FROM    student
WHERE   sno IN
          (SELECT sno
           FROM   sc
           WHERE cno = 'c4');
```

结果如下：



```
SNAME
------
姚二
张三
刘八
```

查询选修 c4 课程的学生学号是一个子查询，查询学生的姓名是父查询。由于可能有多个同学都选修了 c4 课程，所以子查询是一个集合，采用 IN 谓词。

上述查询的执行过程是：先执行子查询，得到选修 c4 课程的学生学号的集合，然后将该集合作为外层父查询的条件，执行父查询，从而得到集合中学号对应的学生姓名。

例题 3.46 查询既没有选修课程 c3，也没有选修课程 c4 的学生学号。

```
SELECT sno
FROM student
WHERE sno NOT IN
            (SELECT sno
            FROM sc
            WHERE cno = 'c3' )
     AND sno NOT IN
               (SELECT sno
               FROM sc
               WHERE cno = 'c4');
```

结果如下：

```
SNO
--------
11432001
11432004
11432005
11432006
11432009
11432010
```

例题 3.47 查询选修了课程名为“证据学”的学生学号和姓名。

```
SELECT sno, sname
FROM student
WHERE sno IN
          (SELECT sno
          FROM sc
          WHERE cno IN
                   (SELECT cno
                   FROM course
                   WHERE cname = '证据学'));
```

结果如下：

```
SNO      SNAME
-------- ------
11432002 姚二
11432008 刘八
11432003 张三
```

此题也可以采用多表连接方法来实现。

```
SELECT student.sno,sname
FROM   student,sc,course
WHERE  student.sno = sc.sno AND course.cno = sc.cno
      AND cname = '证据学';
```

结果如下：

```
SNO      SNAME
-------- ------
11432002 姚二
11432003 张三
11432008 刘八
```

2）带有比较运算符的子查询

带有比较运算符的子查询是指父查询与子查询之间用比较运算符进行连接。只有当内层查询返回的是单值时，才可以用＞、＜、＝、＞＝、＜＝、！＝或＜＞等比较运算符。

例题 3.48　查询与学号“11432007”学生在同一系别的学生学号、姓名和系别名称。

```
SELECT sno,sname,dept
FROM student
WHERE dept = (SELECT dept
             FROM student
             WHERE sno = '11432007');
```

结果如下：

```
SNO      SNAME                DEPT
-------- -------------------- ----------
11432007 陈七                 公安信息系
11432008 刘八                 公安信息系
```

也可以用前面介绍的 IN 谓词来实现。

```
SELECT sno,sname,dept
FROM student
WHERE dept IN (SELECT dept
              FROM student
              WHERE sno = '11432007');
```

结果如下：

```
SNO      SNAME                DEPT
-------- -------------------- ----------
11432007 陈七                 公安信息系
11432008 刘八                 公安信息系
```

注意：当子查询的结果是单个值时，谓词 IN 和“＝”的作用是等价的。当子查询的结果是多个值时，只能够用谓词 IN，而不能用“＝”了。

3）带有 ANY 谓词或 ALL 谓词的子查询

使用 ANY 或 ALL 谓词前必须同时使用比较运算符，含义如表 3.7 所示。

表 3.7 ANY 和 ALL 谓词的使用含义

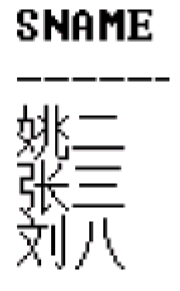
ANY 或 ALL 谓词前的比较运算符	含 义
> ANY	大于子查询结果集中的某个值
> ALL	大于子查询结果集中的所有值
< ANY	小于子查询结果集中的某个值
< ALL	小于子查询结果集中的所有值
>= ANY	大于等于子查询结果集中的某个值
>= ALL	大于等于子查询结果集中的所有值
<= ANY	小于等于子查询结果集中的某个值
<= ALL	小于等于子查询结果集中的所有值
= ANY	等于子查询结果集中的某个值
= ALL	等于子查询结果集中的所有值(无意义)
<> ANY	不等于子查询结果集中的某个值(无意义)
<> ALL	不等于子查询结果集中的任何一个值

注意：<>ALL 等价于 NOT IN；=ANY 等价于 IN；=ALL、<>ANY 没有意义。

例题 3.49 查询选修课程号为 c4 的学生姓名。(与例题 3.45 相同，IN 与 = ANY 等价。)

```
SELECT sname
FROM   student
WHERE sno = ANY
            (SELECT sno
             FROM   sc
             WHERE cno = 'c4');
```

结果如下：



若此题换成查询没有选修课程号为 c4 的学生姓名，则只需将 = ANY 换成<>ALL。因为<>ALL 与 NOT IN 等价。

例题 3.50 查询比所有男同学年龄都大的女同学的学号、姓名和年龄。

```
SELECT sno, sname, age
FROM student
WHERE sex = '女' AND age > ALL(SELECT age
```

```
              FROM student
              WHERE sex = '男');
```

结果如下：

```
SNO      SNAME                AGE
-------- -------------------- ----------
11432007 陈七                  23
```

用聚集函数实现子查询通常比直接用 ANY 或 ALL 谓词查询效率高，ANY 或 ALL 谓词与聚集函数的对应关系如表 3.8 所示。

表 3.8　ANY 或 ALL 谓词与聚集函数的对应关系

比较运算符	ANY	ALL
=	IN	无意义
<>	无意义	NOT IN
<	< MAX	< MIN
<=	<= MAX	<= MIN
>	> MIN	> MAX
>=	>= MIN	>= MAX

例题 3.51　查询其他系中比刑事技术系某一学生年龄大的学生姓名和年龄。

方法一：

```
SELECT sname,age
FROM   student
WHERE  dept <>'刑事技术系'
       AND age > ANY (SELECT age
                      FROM student
                      WHERE dept = '刑事技术系');
```

结果如下：

```
SNAME                AGE
-------------------- ----------
陈七                  23
孙十                  21
刘八                  21
姚二                  20
```

方法二：

```
SELECT sname,age
FROM student
WHERE dept <>'刑事技术系'
      AND age > (SELECT MIN(age)
                 FROM student
                 WHERE dept = '刑事技术系');
```

结果如下：

SNAME	AGE
姚二	20
陈七	23
刘八	21
孙十	21

例题 3.52 查询其他系中比刑事技术系所有学生年龄都大的学生姓名和年龄。

方法一：

```
SELECT sname,age
FROM   student
WHERE  age > ALL (SELECT age
                  FROM student
                  WHERE dept = '刑事技术系');
```

结果如下：

SNAME	AGE
陈七	23

方法二：

```
SELECT sname,age
FROM student
WHERE dept <>'刑事技术系'
      AND age > (SELECT MAX(age)
                 FROM student
                 WHERE dept = '刑事技术系');
```

结果如下：

SNAME	AGE
陈七	23

2. 相关子查询

前面介绍的子查询都是不相关子查询，不相关子查询比较简单，在整个过程中子查询只执行一次，并且把结果用于父查询，即子查询不依赖于外层父查询。而更复杂的情况是子查询要多次执行，子查询的查询条件依赖于外层父查询的某个属性值，这类查询称为相关子查询。

相关子查询的特点：

（1）先执行父查询，后执行子查询。

（2）子查询不能独立运行，子查询的条件依赖外层父查询中取的值。

（3）子查询多次运行。

1）带有比较运算符的相关子查询

例题 3.53　查询所有课程成绩均及格的学生学号和姓名。

```
SELECT sno,sname
FROM   student
WHERE  60<= (SELECT MIN(grade)
              FROM sc
              WHERE student.sno = sc.sno);
```

结果如下：

SNO	SNAME
11432001	陈一
11432003	张三
11432004	李四
11432007	陈七
11432009	张九
11432008	刘八
11432010	孙十

2）有 EXISTS 谓词的子查询

在相关子查询中经常使用 EXISTS 谓词。带有 EXISTS 谓词的子查询不返回任何数据，只产生逻辑真值 true 或逻辑假值 false。

若内层查询结果非空，则外层的 WHERE 子句返回真值。

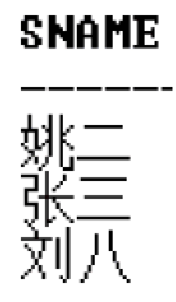
若内层查询结果为空，则外层的 WHERE 子句返回假值。

由 EXISTS 引出的子查询，其目标列表达式通常都用 * ，因为带 EXISTS 的子查询只返回真值或假值，给出列名无实际意义。

例题 3.54　查询选修课程号为 c4 的学生姓名。（与例 3.45 和例 3.49 相同。）

```
SELECT sname
FROM   student
WHERE EXISTS
      (SELECT   *
      FROM   sc
      WHERE student.sno = sc.sno AND cno = 'c4');
```

结果如下：



执行过程：首先取外层查询中 student 表的第一行元组，根据它与内层查询相关属性值(sno)来处理内层查询，若内层查询结果非空，则 EXISTS 为真，就把 student 表的第一行元

组中 sname 值取出放入查询结果的结果集中；然后取 student 表的第二行、第三行……重复上述过程，直到 student 表中所有行全部被检索完为止。

与 EXISTS 谓词相对应的是 NOT EXISTS 谓词。

若内层查询结果非空，则外层的 WHERE 子句返回假值。

若内层查询结果为空，则外层的 WHERE 子句返回真值。

例题 3.55 查询没有选修课程号为 c4 的学生姓名。

```
SELECT sname
FROM   student
WHERE NOT EXISTS
          (SELECT   *
           FROM   sc
           WHERE student.sno = sc.sno AND cno = 'c4');
```

结果如下：

```
SNAME
------
陈一
李四
王五
赵六
陈七
张九
孙十
```

例题 3.56 查询没有选课的学生学号和姓名。

```
SELECT sno, sname
FROM   student
WHERE NOT EXISTS
         (SELECT *
          FROM sc
          WHERE student.sno = sc.sno);
```

结果如下：

```
SNO        SNAME
---------- ------
11432006   赵六
```

例题 3.57 查询所有课程成绩均大于 80 分的学生学号和姓名。

```
SELECT sno, sname
FROM    student
WHERE   sno IN
        (SELECT sno
         FROM sc
```

```
      WHERE  NOT EXISTS
             (SELECT  *
             FROM   sc
             WHERE student.sno = sc.sno AND grade <= 80));
```

结果如下：

```
SNO       SNAME
--------- ------
11432004 李四
11432009 张九
```

为了防止没有选课的学生 11432006 赵六出现在结果集里，所以上例中用到了两层嵌套查询。

3.3.8 集合查询

若把多个 SELECT 语句的结果合并为一个结果集，可用集合操作来完成。集合操作主要包括并操作 UNION、交操作 INTERSECT 和差操作 MINUS。参加集合操作的各个结果表的列数必须相同，对应项的数据类型也必须相同。各个结果表中的列名可以不同。

1. 并操作

SQL 语言使用并操作 UNION 把查询的结果合并起来，并且去掉重复的元组。

例题 3.58 查询侦查系和公安信息系的学生姓名的并集。

```
SELECT sname
FROM student
WHERE dept = '侦查系'
UNION
SELECT sname
FROM student
WHERE dept = '公安信息系';
```

结果如下：

```
SNAME
------
陈七
陈一
刘八
姚二
张三
```

上述集合查询语句的结果等价于：

```
SELECT sname
FROM student
```

```
WHERE dept = '侦查系'OR dept = '公安信息系';
```

2. 交操作

SQL 语言使用 INTERSECT 把同时出现在两个查询的结果取出，实现交操作，并且也会去掉重复的元组。

例题 3.59 查询治安管理系的学生和年龄大于 20 岁的学生的交集。

```
SELECT *
FROM student
WHERE dept = '治安管理系'
INTERSECT
SELECT *
FROM student
WHERE age > 20;
```

结果如下：

SNO	SNAME	SEX	AGE	DEPT
11432010	孙十	女	21	治安管理系

上述集合查询语句的结果等价于：

```
SELECT *
FROM student
WHERE dept = '治安管理系' AND age > 20;
```

3. 差操作

SQL 语言使用 MINUS 把出现在第一个查询结果中，但不出现在第二个查询结果中的元组取出，实现差操作。

例题 3.60 查询治安管理系的学生和年龄大于 20 岁的学生的差集。

```
SELECT *
FROM student
WHERE dept = '治安管理系'
MINUS
SELECT *
FROM student
WHERE age > 20;
```

结果如下：

SNO	SNAME	SEX	AGE	DEPT
11432009	张九	女	18	治安管理系

上述集合查询语句的结果等价于：

```
SELECT *
FROM student
WHERE dept = '治安管理系' AND age <= 20;
```

3.4　数据操纵

3.4.1　插入数据

当基本表建立以后,就可以使用 INSERT 语句向表中插入数据了。INSERT 语句有两种插入形式:插入单个元组和插入多个元组(插入子查询结果)。

1. 插入单个元组

向基本表中插入数据的语法格式如下:

```
INSERT INTO <基本表名> [(<列名 1>,<列名 2>, … ,<列名 n>)]
VALUES(<列值 1>,<列值 2>, … ,<列值 n>)
```

其中,<基本表名>指定要插入元组的表的名字;<列名 1>,<列名 2>,…,<列名 n>为要添加列值的列名序列;VALUES 后则一一对应要添加列的输入值。

注意:

(1) 向表中插入数据之前,表的结构必须已经创建。

(2) 插入的数据及列名之间用逗号分开。

(3) 在 INSERT 语句中列名是可以选择指定的,如果没有指定列名,则表示这些列按表中或视图中列的顺序和个数。

(4) 插入值的数据类型、个数、前后顺序必须与表中属性列的数据类型、个数、前后顺序匹配。

例题 3.61　向学生表中插入一个新的学生记录。

方法一:省略所有列名。

```
INSERT INTO student
VALUES ('11432011','小明','男',20,'侦查系');
```

方法二:指出所有列名。

```
INSERT INTO student(sno,sname,sex,age,dept)
VALUES ('11432011','小明','男',20,'侦查系');
```

两种方法的作用是相同的。

例题 3.62　向学生表中指定的属性列插入数据。

```
INSERT INTO student(sno,sname,sex)
VALUES ('11432012','小强','女');
```

其中,没有插入数据的属性列的值均为空值。

注意:在向表中插入数据时,所插入的数据应满足定义表时的约束条件。例如,如果再次向 student 表中插入学号为 11432012 的学生记录时,系统就会给出错误提示信息,违反了主键约束。如果再插入另一个新的学号为 11432013 的学生记录,但不知道此同学的性别,插入的性别属性列的值为空值,此时系统也会给出错误提示信息,违反了定义表时对于"性别"字段的非空约束。

2. 插入多个元组

向基本表中插入数据的语法格式如下:

```
INSERT INTO <基本表名> [(<列名 1>,<列名 2>, … ,<列名 n>)] 子查询;
```

如果列名序列省略则子查询所得到的数据列必须和要插入数据的基本表的数据列完全一致。如果列名序列给出则子查询结果与列名序列要一一对应。

例题 3.63 如果已经创建了课程平均成绩记录表 course_avg(cno,ave),其中 ave 表示每门课程的平均成绩,向 course_avg 表中插入每门课程的平均成绩。

```
INSERT INTO course_avg(cno,ave)
SELECT cno,AVG(grade)
FROM sc
GROUP BY cno;
```

3.4.2 修改数据

如果表中的数据出现错误,可以利用 UPDATE 命令进行修改。UPDATE 语句用以修改满足指定条件的元组信息。满足指定条件的元组可以是一个元组,也可以是多个元组。UPDATE 语句的一般语法格式为:

```
UPDATE <基本表名>
SET <列名 1> = <表达式> [,<列名 2> = <表达式>] …
[WHERE <条件>];
```

其中,UPDATE 关键字用于定位修改哪一张表,SET 关键字用于定位修改这张表中的哪些属性列,WHERE<条件>用于定位修改这些属性列当中的哪些行。

UPDATE 语句只能修改一个基本表中满足 WHERE<条件>的元组的某些列值,即其后只能有一个基本表名。这里,WHERE<条件>是可选的,如果省略不选,则表示要修改表中所有的元组。

1. 修改某一个元组的值

例题 3.64 将高等数学课程的学分改为 4 学分。

```
UPDATE course
SET credit = 4
```

```
WHERE cname = '高等数学';
```

2. 修改多个元组的值

例题 3.65　将所有男同学的年龄增加 2 岁。

```
UPDATE student
SET age = age + 2
WHERE sex = '男';
```

例题 3.66　将所有课程的学分减 1。

```
UPDATE course
SET credit = credit - 1;
```

3. 带子查询的更新

在 UPDATE 语句中可以嵌套子查询，用于构造修改的条件。

例题 3.67　将所有选修高等数学课程的学生成绩改为 0 分。

```
UPDATE sc
SET grade = 0
WHERE '高等数学' = (SELECT cname
                FROM course
                WHERE course.cno = sc.cno);
```

注意：在修改表中的数据时，修改后的数据应满足定义表时设定的约束条件，否则系统就会给出错误提示信息。例如，如果将某个学生的年龄修改为 14 岁，就违反了表定义时对于“年龄”字段的检查约束。

3.4.3　删除数据

如果不再需要学生选课系统中的某些数据，此时应该删除这些数据，以释放其所占用的存储空间。

DELETE 语句的一般语法格式为：

```
DELETE FROM <表名> [WHERE <条件>];
```

DELETE 语句的功能是从指定表中删除满足 WHERE<条件>的所有元组。DELETE 语句只删除表中的数据，而不能删除表的结构，所以表的定义仍然在数据字典中。如果省略 WHERE<条件>，表示删除表中全部的元组信息。

1. 删除某一个元组的值

例题 3.68　删除学号为 11432011 的学生记录。

```
DELETE FROM student
WHERE sno = '11432011';
```

2. 删除多个元组的值

例题 3.69 删除学号为 11432005 学生的选课记录。

```
DELETE FROM sc
WHERE sno = '11432005';
```

每一个学生可能选修多门课程，所以 DELETE 语句会删除这个学生的多条选课记录。

例题 3.70 删除所有学生的选课记录。

```
DELETE FROM sc;
```

3. 带子查询的删除

在 DELETE 语句中同样可以嵌套子查询，用于构造删除的条件。

例题 3.71 删除王五同学的选课记录。

```
DELETE FROM sc
WHERE '王五' = ( SELECT sname
                FROM   student
                WHERE student.sno = sc.sno);
```

注意：在删除表中的数据时，应满足定义表时设定的约束条件，否则系统会给出错误提示信息。例如，如果想删除学生表 student 中的某一个学生记录，但这个学生在选课表 sc 中存在选课记录，此时删除学生记录的操作就会出错，因为外键关联的表的数据删除顺序是先删除从表中的数据，再删除主表中的数据。所以，正确的做法是先从选课表 sc 中将这个学生的选课记录删除，再从学生表 student 中删除这个学生的记录。

3.5 视　　图

视图是从一个或几个基本表(或视图)导出的表，它与基本表不同，是一个虚表。视图一经定义，就可以和基本表一样被查询、被删除，也可以在一个视图之上再定义新的视图，但对视图的更新(增加、删除、修改)操作则有一定的限制。

视图的特点如下：

(1) 视图是从现有的一个或多个表中提取出来的，可以屏蔽表中的某些信息。

(2) 视图是一个虚表，对视图的操作实际上是对基本表的操作。

(3) 数据库中只存放视图的定义，不存放视图对应的数据。这些数据仍存放在原来的基本表中，所以基本表中的数据发生变化，从视图中查询的数据也就随之改变了。

(4) 视图可以简化用户查询操作，隐蔽表之间的连接。

3.5.1 定义视图

建立视图的一般语法格式如下：

```
CREATE VIEW <视图名>[(<列名>[,<列名>]…)]AS (子查询)
[WITH CHECK OPTION]
[WITH READ ONLY];
```

其中：

(1) 视图中的列名序列要么全部指定，要么全部省略。当列名序列省略时，直接使用子查询 SELECT 子句里的各列名作为视图列名。

下列几种情况不能省略列名序列：

① 多表连接时选出了几个同名列作为视图的字段。

② 视图列名中有常数、聚集函数或列表达式。

③ 需要用更合适的新列名作视图列的列名。

(2) WITH CHECK OPTION 是可选项，该选项表示对所建视图进行 INSERT、UPDATE 和 DELETE 操作时，系统需检查该操作的数据是否满足子查询中 WHERE 子句里限定的条件，若不满足，则系统拒绝执行。

(3) WITH READ ONLY 是可选项，该选项保证在视图上不能进行任何 DML 操作。

例题 3.72　建立侦查系学生的视图，包括学号、姓名、性别和年龄。并要求进行插入和修改操作时仍要保证此视图中只有侦查系的学生。

```
CREATE VIEW invest_student
AS
SELECT sno, sname, sex, age
FROM student
WHERE dept = '侦查系'
WITH CHECK OPTION;
```

本例中，视图列名及顺序与 SELECT 子句中一样，所以视图名 invest_student 后列名被省略。对所建视图进行 INSERT、UPDATE 和 DELETE 操作时，系统自动检查该操作的数据是否满足计算机系学生的条件，若不满足，则系统拒绝执行。

例题 3.73　建立侦查系学生的只读视图，包括学号、姓名、性别和年龄。

```
CREATE VIEW invest_student_only
AS
SELECT sno, sname, sex, age
FROM student
WHERE dept = '侦查系'
WITH READ ONLY;
```

本例中，视图 invest_student_only 一旦建立，就不允许在视图上进行任何 DML 操作。

例题 3.74　建立侦查系选修高等数学课程的学生视图，包括学号、姓名和成绩。

```
CREATE VIEW invest_student_maths
AS
```

```
SELECT student.sno,sname,grade
FROM   student,course,sc
WHERE  student.sno = sc.sno AND course.cno = sc.cno
       AND dept = '侦查系' AND cname = '高等数学';
```

本例中，视图 invest_student_maths 是从多张基本表中提取出来的，所以，不能对视图 invest_student_maths 进行插入、更新和删除操作。

视图不仅可以建立在一个或多个基本表上，也可以建立在一个或多个已经定义好的视图上，或建立在基本表与视图上。

例题 3.75 建立侦查系年龄大于 18 岁的学生视图，包括学号和姓名。

```
CREATE VIEW invest_student_age
AS
SELECT sno,sname
FROM invest_student
WHERE age > 18;
```

本例中，视图 invest_student_age 是从已经创建的视图 invest_student 中提取出来的。有时，还可以用带有聚集函数和 GROUP BY 子句的查询来定义视图。

例题 3.76 建立一个记录每个系别学生人数的视图，包括系别名称和学生人数。

```
CREATE VIEW dept_count(dept,num)
AS
SELECT dept,COUNT(sno)
FROM student
GROUP BY dept;
```

本例中，由于 AS 子句中 SELECT 语句的目标列学生人数是通过使用聚集函数得到的，所以 CREATE VIEW 中必须明确定义组成 dept_count 视图的各个属性列名，必须使用列别名来命名表达式 COUNT(sno)。

3.5.2 查询视图

视图是从一个或多个表中导出的虚表，具有表的基本特性。从用户角度来说，基于视图的数据查询与基于基本表的数据查询一样使用 SELECT 语句，查询视图的方法与查询基本表的方法一致，所以 DBMS 执行对视图的查询实际上是根据视图的定义转换成等价的对基本表的查询。

例题 3.77 在侦查系学生的视图中查找男同学的信息。

```
SELECT *
FROM invest_student
WHERE sex = '男';
```

DBMS 对某 SELECT 语句进行处理时，若发现被查询对象是视图，则 DBMS 将进行如

下操作：

(1) 从数据字典中取出视图的定义。

(2) 把视图定义的子查询和本 SELECT 语句定义的查询相结合，生成等价的对基本表的查询(此过程称为视图的消解)。

(3) 执行对基本表的查询，把查询结果(作为本次对视图的查询结果)向用户显示。

因此，本例转换后的查询语句为：

```
SELECT sno, sname, sex, age
FROM student
WHERE dept = '侦查系' AND sex = '男';
```

通常，对视图的查询是不会出现问题的。但有时视图消解过程不能给出语法正确的查询条件，可能不是查询语句的语法错误，而是转换后的语法错误。此时，用户需要自己把对视图的查询转化为对基本表的查询。

3.5.3　操纵视图

操纵视图是指通过视图来插入(INSERT)、删除(DELETE)和修改(UPDATE)数据。同查询视图一样，由于视图是不实际存储数据的虚表，因此对视图的操纵，最终要转换为对基本表的操纵。

此外，用户通过视图操纵数据不能保证被操纵的数据符合原来视图中定义的 AS<子查询>的条件。因此，在定义视图时，若加上子句 WITH CHECK OPTION，则在对视图操纵时，系统将自动检查先前定义时的条件是否满足。若不满足，则拒绝执行。

1. 对视图的数据插入

例题 3.78　建立公安信息系学生的视图，包括学号、姓名、性别和系别名称。向公安信息系学生的视图中插入一个新的学生记录，其中学号为 11432013，姓名为"小文"，性别为"女"，系别为"公安信息系"。

视图的建立：

```
CREATE VIEW is_student
AS
SELECT sno, sname, sex, dept
FROM student
WHERE dept = '公安信息系';
```

视图的数据插入：

```
INSERT INTO is_student
VALUES('11432013', '小文', '女', '公安信息系');
```

上述语句在执行时，将转换成对学生表 student 的数据插入：

```
INSERT INTO student
VALUES('11432013','小文','女',NULL,'公安信息系');
```

2. 对视图的数据修改

例题 3.79 将公安信息系学生的视图中,学号为11432013的学生姓名改为"周文"。

```
UPDATE is_student
SET sname = '周文'
WHERE sno = '11432013';
```

上述语句在执行时,将转换成对学生表student的数据修改:

```
UPDATE student
SET sname = '周文'
WHERE sno = '11432013' AND dept = '公安信息系';
```

3. 对视图的数据删除

例题 3.80 从公安信息系学生的视图中,删除学号为11432013的学生记录。

```
DELETE FROM is_student
WHERE Sno = '11432013';
```

上述语句在执行时,将转换成对学生表student的数据删除:

```
DELETE FROM student
WHERE Sno = '11432013' AND dept = '公安信息系'
```

并不是所有的视图操纵都能转换成有意义的对基本表的操纵。为了能正确执行视图操纵,各DBMS对视图操纵都有若干规定,由于各系统实现方法上的差异,这些规定也不尽相同。一般的规定如下:

(1) 通常对于由一个基本表导出的视图,如果是从基本表中去掉除码外的某些列和行,是允许操纵的。

(2) 若视图是两个以上的基本表导出的,则此视图不允许操纵。

(3) 若视图的列是由聚集函数或计算列构成的,则此视图不允许操纵。

(4) 若视图定义中含有DISTINCT、GROUP BY等子句,则此视图不允许操纵。

3.5.4 删除视图

删除视图的一般语法格式如下:

```
DROP VIEW <视图名>;
```

注意:

(1) 删除视图后,视图的定义将从数据字典中删除,但基本表中的数据不受影响。

(2) 删除基本表后,由该基本表导出的所有视图并没有被删除,但均已无法使用。

例题 3.81　删除公安信息系学生的视图。

```
DROP VIEW is_student;
```

3.5.5　视图的优点

视图作为数据库中的一个重要的概念,有很多的优点,主要包括以下几个方面:

(1) 为用户集中数据,简化用户的数据查询和处理。有时用户所需要的数据分散在多个表中,定义视图可将它们集中在一起,从而方便用户的数据查询和处理。

(2) 屏蔽数据库的复杂性。用户不必了解复杂的数据库中表的结构,并且数据库中表的更改也不影响用户对数据库的使用。

(3) 简化用户权限的管理。只需授予用户使用视图的权限,不必指定用户只能使用表的某些特定列,增加了安全性。

(4) 便于数据共享。各用户不必都定义和存储自己所需的数据,可共享数据库的数据,同样的数据只需存储一次。

(5) 可以重新组织数据,以便输出到其他的应用程序中。

3.6　实　　验

3.6.1　实验 1　SQL * PLUS 常用命令练习

1. 实验目的

(1) 掌握 Oracle 客户端工具 SQL * PLUS 的交互运用。

(2) 熟悉 SQL * PLUS 中的常用命令。

2. 实验内容

(1) 以 system 用户身份登录 SQL * PLUS,登录后显示当前用户。

```
SHOW USER;
```

(2) 查看 system 用户下的表。

```
SELECT table_name FROM user_tables;
```

(3) 查看员工表 emp 的结构。

```
DESC emp;
```

(4) 查看 SQL * PLUS 里的命令。

```
HELP INDEX;
```

(5) 查看 RUN 命令的使用方法及简写形式。

```
? RUN;
```

(6) 设置行宽和列宽。
(设置前与设置后分别运行 SELECT * FROM emp;看结果变化。)

```
SET LINESIZE 200;
SET PAGESIZE 200;
```

(7) 查找缓存区内最近写过的命令。

```
list;
```

(8) 执行缓存区里的命令。

```
/ , run ,r;
```

(9) 替换命令。

```
CHANGE/old/new
```

将当前行中的 old 替换为 new。
(10) 编辑命令:对当前的输入进行编辑。

```
EDIT; (Windows 默认在记事本中编辑)
```

(11) 保存最近写过的命令。

```
SAVE c:\part1; (默认保存成.sql)
SAVE c:\part1.txt;
```

(12) 读入命令。

```
GET c:\part1.sql;
```

(13) 读入并执行。

```
START c:\part1.sql;
```

(14) 保存所有的操作。

```
SPOOL c:\part2.sql;(先创建文本,从想保存的位置开始)
SELECT * FROM emp;(写入想保存的命令,包括结果)
SPOOL OFF;(操作结束的位置)
```

3. 考核标准

本实验为选做实验,根据课时进度安排,既可以在课堂上完成,也可作为学生课外作业独立完成。要求学生在自己的 PC 上成功安装 Oracle,并且能够熟练使用 SQL * PLUS 常

用命令进行练习即为优秀。如果出现错误，则根据错误情况灵活给分。

3.6.2　实验 2　数据定义语言 DDL

1. 实验目的

(1) 掌握基本表的创建方法。

(2) 掌握基本表结构的修改方法。

(3) 掌握基本表删除的方法。

2. 实验内容

(1) 按要求采用不同的约束类型创建科室表和医生表。

① 科室表：dept(deptno,dname,loc)，表中属性列依次是科室编号、科室名称、科室所在地点。科室表结构如表 3.9 所示。

表 3.9　科室表结构

列名	数据类型	长度	完整性约束
deptno	CHAR	10	主键
dname	VARCHAR	15	唯一
loc	VARCHAR	20	无

② 医生表：doctor(docno,docname,age,sal,deptno)，表中属性列依次是医生编号、医生姓名、年龄、工资、所在科室编号。医生表结构如表 3.10 所示。

表 3.10　医生表结构

列名	数据类型	长度	完整性约束
docno	CHAR	10	主键
docname	VARCHAR	15	非空
age	INT	无	年龄在 18～60 岁之间
sal	NUMBER	无	无
deptno	CHAR	10	外键(参照 dept 表中 deptno)

(2) 按要求对于表的结构进行修改。

① 对表增加一列。

在医生表中增加一个属性列，birthday(生日)，数据类型是 DATE。

② 改变列的类型。

将科室表中 dname 属性列的类型改为 VARCHAR2(20)。

③ 增加约束条件。

在医生表中添加一个名为 CHK_SAL 的约束，从而保证医生工资的取值总是在 1000 到 8000 之间，即 sal BETWEEN 1000 AND 8000。

④ 删除原有的列。

删除医生表中 birthday 属性列。

(3) 按要求删除基本表。

删除医生表 doctor。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相应实验内容。程序语句无语法错误,书写规范,运行结果正确为优秀。如果出现错误,根据错误点数以及难易程度灵活给分。

3.6.3 实验 3 数据操纵语言 DML

1. 实验目的

(1) 掌握基本表中数据的插入操作。

(2) 掌握基本表中数据的更新操作。

(3) 掌握基本表中数据的删除操作。

2. 实验内容

(1) 创建教师信息基本表。

教师信息表:teacher(tno,tname,sex,sal,tdept),表中属性列依次是教师编号、教师姓名、性别、工资和系别名称。教师信息表的结构如表 3.11 所示。

表 3.11 教师信息表

列名	数据类型	长度	完整性约束
tno	CHAR	8	主键
tname	VARCHAR2	20	非空
tsex	VARCHAR2	6	无
tsal	NUMBER	无	工资大于 1800
tdept	CHAR	20	无

(2) 练习向基本表中插入数据、修改数据和删除数据。

① 向教师表中插入如表 3.12 所示的数据。

表 3.12 向教师表中插入数据

tno	tname	tsex	tsal	tdept
T001	张老师	女	3000	侦查系
T002	王老师	男	2800	侦查系
T003	李老师			公安信息系
T004	张老师	男	3500	公安信息系
T005	刘老师	女	2200	治安管理系

② 将“公安信息系”更名为“信息系”。

③ 将王老师的工资更改为 3300。

④ 删除教师表中所有侦查系的教师信息。

⑤ 删除教师表中的全部数据。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，按照实验步骤完成相应实验内容。程序语句无语法错误，书写规范，运行结果正确为优秀。如果出现错误，则根据错误点数以及难易程度灵活给分。

3.6.4　实验 4　单表查询

1. 实验目的

(1) 掌握对于单个基本表的数据查询方法。

(2) 掌握聚集函数的应用。

2. 实验内容

(1) 创建员工信息表。

员工信息表：employees(eno,ename,sex,age,job,sal,dept)，表中属性列依次是员工编号、员工姓名、性别、年龄、工作岗位、工资和部门名称。员工信息表的结构如表 3.13 所示。

表 3.13　员工信息表

列名	数据类型	长度	完整性约束
eno	CHAR	8	主键
ename	VARCHAR2	10	非空
sex	CHAR	6	取值只允许为男或女
age	INT	无	年龄大于 18 岁
job	VARCHAR2	20	无
sal	NUMBER	无	无
dept	VARCHAR2	20	无

(2) 向已创建的员工表中插入如表 3.14 所示的数据。

表 3.14　向员工表中插入数据

eno	ename	sex	age	job	sal	dept
1001	张三	男	20	销售	1000	市场部
1002	李四	女	26	会计	1600	财务部
1003	王五	女	22	销售	1000	市场部
1004	赵六	男	19			
1005	张七	女	23	测试	1400	技术部
1006	赵八	男	30	研发	2000	技术部

(3) 按要求完成各种单表信息查询,并验证聚集函数的功能。

① 查询所有员工的姓名、性别和工资。

② 查询员工表中所有的部门名称(要求去掉重复的值)。

③ 查询技术部员工的姓名和出生年份。

④ 查询工资超过 1200 元的员工的姓名和年龄。

⑤ 查询年龄不在 20~25 岁之间的员工的姓名和工资。

⑥ 查询财务部、技术部的员工的姓名和性别。

⑦ 查询所有姓张的员工的姓名、年龄和工作。

⑧ 查询工作岗位为空的员工姓名和年龄。

⑨ 查询市场部里年龄小于 25 岁的男员工的姓名。

⑩ 查询年龄超过 20 岁的员工的姓名和工资,查询结果按照工资的降序排列。

⑪ 查询市场部的员工人数。

⑫ 查询公司中员工的最高工资。

⑬ 查询公司中员工的最低工资。

⑭ 查询技术部门员工的平均年龄。

⑮ 查询市场部中员工的工资总额。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相应实验内容。程序语句无语法错误,书写规范,运行结果正确为优秀。如果出现错误,根据错误点数以及难易程度灵活给分。

3.6.5 实验 5 多表连接查询和集合查询

1. 实验目的

(1) 掌握多表连接的查询方法。

(2) 了解连接查询中的左外连接和右外连接。

(3) 熟悉集合查询的应用。

2. 实验内容

样本数据库中,学生表 student、课程表 course 和选课表 sc 的数据信息如图 3.4~图 3.6 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 3.4 学生表 student 中的数据

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图 3.5　课程表 course 中的数据

SNO	CNO	GRADE
11432001	c1	75
11432001	c2	95
11432002	c1	82
11432002	c2	88
11432002	c4	76
11432002	c5	55
11432003	c1	65
11432003	c2	72
11432003	c3	90
11432003	c4	85
11432004	c1	93
11432004	c2	96
11432004	c5	91
11432005	c1	50
11432005	c2	70
11432005	c5	88
11432005	c8	43
11432007	c1	75
11432007	c2	73
11432007	c3	66
11432007	c6	82
11432007	c7	94
11432008	c1	82
11432008	c2	77
11432008	c3	85
11432008	c4	87
11432008	c5	82
11432008	c6	94
11432008	c7	92
11432008	c8	89
11432009	c1	86
11432009	c2	
11432009	c8	
11432010	c1	73
11432010	c2	
11432010	c8	76

图 3.6　选课表 sc 中的数据

（1）根据样本数据库中的表和数据，进行多表连接查询操作的练习。

① 查询选修了 c4 号课程的学生的姓名及其成绩，查询结果按成绩降序排列。

② 求男同学的总人数和平均年龄。

③ 查询每名学生的学号、选课门数和平均成绩。

④ 查询平均成绩大于 80 分的学生学号以及平均成绩。

⑤ 查询选修了“JSP 程序设计”课程的学生学号、姓名及其成绩。

⑥ 查询侦查系学生的选课情况，要求列出学生的名字、所选课程的名称和成绩。

(2) 根据样本数据库中的表和数据，进行集合查询操作的练习。

① 查询侦查系和刑事技术系学生的基本信息(集合并运算)。

② 查询公安信息系中选修 c4 课程的学生学号(集合交运算)。

③ 查询治安管理系的学生与年龄不大于 20 岁的学生的差集(集合差运算)。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，按照实验步骤完成相应实验内容。程序语句无语法错误，书写规范，运行结果正确为优秀。如果出现错误，根据错误点数以及难易程度灵活给分。

3.6.6 实验 6 嵌套查询

1. 实验目的

(1) 掌握不相关子查询的查询方法。

(2) 掌握相关子查询的查询方法。

(3) 理解不相关子查询与相关子查询的区别。

2. 实验内容

样本数据库中，学生表 student、课程表 course 和选课表 sc 的数据信息如图 3.7～图 3.9 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 3.7 学生表 student 中的数据

(1) 根据样本数据库中的表和数据，进行不相关子查询的练习。

① 查询与王五同一个系别的学生姓名和年龄。

② 查询选修了“公共安全危机管理”课程的学生学号和姓名。

③ 查询比所有侦查系学生年龄都大的学生的基本情况。

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图 3.8　课程表 course 中的数据

SNO	CNO	GRADE
11432001	c1	75
11432001	c2	95
11432002	c1	82
11432002	c2	88
11432002	c4	76
11432002	c5	55
11432003	c1	65
11432003	c2	72
11432003	c3	90
11432003	c4	85
11432004	c1	93
11432004	c2	96
11432004	c5	91
11432005	c1	50
11432005	c2	70
11432005	c5	88
11432005	c8	43
11432007	c1	75
11432007	c2	73
11432007	c3	66
11432007	c6	82
11432007	c7	94
11432008	c1	82
11432008	c2	77
11432008	c3	85
11432008	c4	87
11432008	c5	82
11432008	c6	94
11432008	c7	92
11432008	c8	89
11432009	c1	86
11432009	c2	
11432009	c8	
11432010	c1	73
11432010	c2	
11432010	c8	76

图 3.9　选课表 sc 中的数据

④ 查询平均成绩最高的学生学号。

⑤ 查询李四同学不学的课程的课程号。

(2) 根据样本数据库中的表和数据,进行相关子查询的练习。

① 在选课信息表中查询选修"证据学"课程的学生学号和成绩。

② 查询没有选修 c2 课程的学生学号和姓名。

③ 查询所有课程成绩均大于 70 分的学生姓名。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相应实验内容。程序语句无语法错误,书写规范,运行结果正确为优秀。如果出现错误,根据错误点数以及难易程度灵活给分。

3.6.7 实验 7 视图

1. 实验目的

(1) 掌握视图的建立方法。

(2) 掌握视图的查询方法。

(3) 掌握视图的删除方法。

2. 实验内容

样本数据库中,学生表 student 的数据信息如图 3.10 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 3.10 学生表 student 中的数据

根据样本数据库中学生表的结构和数据,进行视图操作的练习。

(1) 建立"刑事技术系"学生的视图 crim_student,包括学号、姓名和年龄。

(2) 查询视图 crim_student 中年龄大于 20 岁的学生姓名。

(3) 将视图 crim_student 中学号为 11432004 的学生的姓名改为"李子"。

(4) 删除视图 crim_student 中姓名为"赵六"的学生信息。

(5) 删除视图 crim_student。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相

应实验内容。程序语句无语法错误，书写规范，运行结果正确为优秀。如果出现错误，根据错误点数以及难易程度灵活给分。

3.7　本章小结

本章首先介绍了 SQL 语言的产生、发展和特点。SQL 称为结构化查询语言，在许多关系数据库管理系统中均可使用，其功能并非仅局限于查询，它集数据定义、数据查询、数据操纵、数据控制功能于一体。

其次介绍了 SQL 语言的数据定义功能、数据查询功能和数据操纵功能。数据定义功能包括基本表、索引、视图的创建、修改和删除。数据查询功能是最丰富的，也是最复杂的。它是本章要求重点掌握的内容，包括单表查询、连接查询、嵌套查询、集合查询等。查询语句中可以使用聚集函数完成相关计算，可以使用分组子句将查询结果按某一属性列的值分组，可以使用排序子句将查询结果按指定的属性列进行排序输出。数据操纵功能包括数据插入、数据修改和数据删除操作。

最后介绍了视图，视图是为了确保数据表的安全性和隐蔽性从一个或多个表中或其他视图中使用 SELECT 语句导出的虚表。数据库中仅存放视图的定义，而不存放视图所对应的数据，数据仍然存放在基本表中，对视图中数据的操纵实际上仍是对组成视图的基本表数据的操纵。

3.8　课后习题

一、选择题

1. 在关系数据库中，SQL 是指(　　)。
   A. Selected Query Language
   B. Procedured Query Language
   C. Standard Query Language
   D. Structured Query Language
2. SQL 的运算对象和结果都是(　　)。
   A. 数据　　B. 属性　　C. 关系　　D. 数据项
3. 在创建基本表的过程中，下列说法正确的是(　　)。
   A. 在一个数据库中，两个基本表的名字可以相同
   B. 在给表命名时，第一个字符不能是数字
   C. 表名和属性列的名字区分大小写
   D. 在给表中的属性列命名时，第一个字符必须是字母或数字
4. 下列哪种操作符号可以和 NULL 值进行比较？(　　)

A. IS　　B. =　　C. LIKE　　D. <>

5. 涉及四张表的查询时,WHERE 子句中至少有(　　)个条件表达式。

A. 1　　B. 2　　C. 3　　D. 4

6. 自然连接是关系数据库中重要的关系运算,下列说法正确的是(　　)。

A. 自然连接就是连接,只是说法不同罢了

B. 自然连接其实是等值连接,它与连接不同

C. 自然连接是去掉重复属性的等值连接

D. 自然连接是去掉重复元组的等值连接

7. 在关系数据库中,实现表与表之间的联系是通过(　　)。

A. 实体完整性规则　　B. 参照完整性规则

C. 用户自定义的完整性　　D. 属性的值域

8. 在 SQL 语句中,HAVING 子句用于筛选满足条件的(　　)。

A. 行　　B. 列　　C. 元组　　D. 分组

9. 下列不属于不相关子查询的特点的是(　　)。

A. 可以运行多次

B. 能独立运行,子查询条件不依赖父查询

C. 只能运行一次

D. 先执行子查询,后执行父查询

10. 关于视图,以下说法不正确的是(　　)。

A. 视图是虚表

B. 数据库中只存放视图的定义,不存放视图对应的数据

C. 使用视图可以简化用户的数据查询和处理

D. 使用视图可以加快查询语句的执行速度

二、简答题

1. 建立索引的目的是什么? 是否索引建立得越多越好?

2. 在 SELECT 语句中,HAVING 子句与 WHERE 子句的区别是什么?

3. 什么是基本表? 什么是视图? 两者之间的区别和联系是什么?

三、应用题

1. 某数据库中包含图书信息、读者信息和借阅信息三张基本表。

图书信息表:book(bno,bname,author,price),表中属性列依次是图书编号、图书名称、图书作者和图书价格。

读者信息表:reader(rno,rname,address),表中属性列依次是借书证号、读者姓名和读者地址。

借阅信息表:br(bno,rno,datetime),表中属性列依次是图书编号、借书证号和借书日期。

图书信息表 book 结构如下：

列名	数据类型	长度	完整性约束
bno	CHAR	10	主键
bname	VARCHAR	20	非空
author	VARCHAR	20	无
price	NUMBER	无	大于 0

读者信息表 reader 结构如下：

列名	数据类型	长度	完整性约束
rno	CHAR	10	主键
rname	VARCHAR	20	非空
address	VARCHAR	50	无

借阅信息表 br 结构如下：

列名	数据类型	长度	完整性约束
bno	CHAR	10	外键(参照 book 表中 bno)
rno	CHAR	10	外键(参照 reader 表中 rno)
datetime	DATE	无	无

主键为(bno,rno)；

(1) 用 SQL 语句实现以下基本表的创建：

① 图书信息表(book)的创建。

② 读者信息表(reader)的创建。

③ 借阅信息表(br)的创建。

(2) 根据各表结构，用 SQL 语句完成下列操作：

① 将图书 book 表中图书名称属性列的数据类型改为 varchar(20)。

② 向读者 reader 表中增加年龄属性列 age，数据类型为 number。

③ 将图书编号为 b1 的图书价格改为 24。

④ 删除“张三”的借阅信息。

⑤ 查询图书价格超过 100 元的图书数量。

⑥ 查询书名中含有“数据库”的图书编号和图书价格。

⑦ 查询借阅了“时间简史”图书的读者姓名和借阅时间。

⑧ 查询借阅了 5 本以上图书的读者姓名。

⑨ 查询一本图书都没有借阅的借书证号和读者姓名。

⑩ 查询图书表中最贵的图书名称和价格。

2. 某数据库中包含供应商信息、零件信息、项目信息和供应情况信息四张基本表。

供应商信息表：s(sno,sn,city)，表中属性列依次是供应商编号、供应商名和供应商所在城市。

零件信息表：p(pno,pn,color,weight)，表中属性列依次是零件编号、零件名、颜色和重量。

项目信息表：j(jno,jn,city)，表中属性列依次是项目编号、项目名称和项目所在城市。

供应情况信息表：spj(sno,pno,jno,qty)，表中属性列依次是供应商编号、零件编号、项目编号和供应数量。

供应商信息表 s 结构如下：

列名	数据类型	长度	完整性约束
sno	CHAR	10	主键
sn	VARCHAR	20	非空
city	VARCHAR	20	无

零件信息表 p 结构如下：

列名	数据类型	长度	完整性约束
pno	CHAR	10	主键
pn	VARCHAR	20	非空
color	VARCHAR	20	无
weight	NUMBER	无	大于 0

项目信息表 j 结构如下：

列名	数据类型	长度	完整性约束
jno	CHAR	10	主键
jn	VARCHAR	20	非空
city	VARCHAR	20	无

供应情况信息表 spj 结构如下：

列名	数据类型	长度	完整性约束
sno	CHAR	10	外键(参照 s 表中 sno)
pno	CHAR	10	外键(参照 p 表中 pno)
jno	CHAR	10	外键(参照 j 表中 jno)
qty	INT	无	无

主键为(sno,pno,jno)；

（1）用 SQL 语句实现以下基本表的创建：
① 供应商信息表(s)的创建。
② 零件信息表(p)的创建。
③ 项目信息表(j)的创建。
④ 供应情况信息表(spj)的创建。
（2）根据各表结构，用 SQL 语句完成下列操作：
① 查询所有零件的名称、颜色和重量。
② 查询所在城市为“大连”的所有项目的详细信息。
③ 查询重量最轻的零件名称。
④ 查询为编号为 j1 的项目提供零件的供应商名称。
⑤ 查询由编号为 s1 的供应商提供的零件颜色。
⑥ 查询编号为 j2 的项目使用的各种零件名称及其数量。
⑦ 查询供应绿色的零件编号为 p2 且供应数量超过 500 的供应商名称。
⑧ 查询每个城市的城市名称及供应商数量，查询结果按照数量的降序排列。
⑨ 查询没有使用“大连”产的零件的项目编号。
⑩ 查询为编号 j1 和 j2 的项目提供零件的供应商编号。

第 4 章

数据库设计和规范化理论

4.1 关系数据库规范化理论

4.1.1 问题引入

对于任何管理信息系统的应用软件开发而言，其核心技术都要涉及数据库设计方面的知识，而要设计出一个性能良好的数据库应用系统并非一件简单的工作。关系数据库设计是对数据进行组织化和结构化的过程，核心问题是关系模型的设计。关系模式设计的好坏将直接影响到数据库设计的成败，而关系规范化理论则是指导关系模式设计的标准。

前面已经介绍了关系模型、关系数据库的基本概念。关系数据库是由实体与实体之间联系的关系集合构成的。关系数据库设计理论所要研究的就是针对某一个具体问题，确定如何构造一个适合于它的数据模式，即构造几个关系模式，每个关系模式应该由哪些属性组成等。

下面通过实例来说明采用不同的数据库模式将产生不同的效果。

例如，某学校要建立一个数据库以描述学生选修课程的情况。由现实世界的已知事实可以得到如下对应关系：每一名学生可以选修多门课程，每一门课程可以被多名学生所选修；每一名学生选修一门课程都会有一个成绩。

针对上述情况可能设计出以下两种关系模式。

1. 只产生一个关系模式

学生选课关系模式(学号，姓名，性别，年龄，系别，课程号，课程名，教师名，学分，成绩)。

2. 产生三个关系模式

学生关系模式(学号，姓名，性别，年龄，系别)；

课程关系模式(课程号，课程名，教师名，学分)；

选课关系模式(学号，课程号，成绩)。

比较分析这两种关系模式，发现第一种设计方法可能带来如下问题。

1. 数据冗余

当每一个学生选修多门课程的时候，这个学生的姓名、性别、年龄和所在系别是被重复存储的，这种重复存储是毫无意义的，浪费了大量的存储器资源，是数据冗余。

2. 修改异常

由于数据冗余，当修改某些属性（如学生的年龄）时，可能有一部分相关元组被修改，而另一部分相关元组没有被修改（同一个学生可能对应两个年龄），这就造成了数据的不一致性。

3. 插入异常

第一个关系模式中的主码是（学号，课程号）的属性组合，假如要插入刚入学的大一新生的信息，学号为 11432011，姓名为周三，女，18 岁，侦查系；由于新生刚入学还未选课，选修课程号为空。此时，则无法将这条信息插入学生选课关系模式中。因为在插入数据时，主码是不允许为空的，而这时主码的一部分（课程号）为空，因而导致学生信息插入不成功。

4. 删除异常

如果只有陈七同学选修了“证据学”课程，那么在陈七同学毕业离校的时候，学校在删除陈七同学基本信息的同时，也将“证据学”这门课程的基本信息彻底删除了，丢失了应该保存的课程信息。

由于存在上述问题，显然第一种设计不是好的关系模式。第二种设计方法就不存在上述问题，消除了数据冗余，消除了修改、插入、删除异常。但这种方法也有自己的缺点，查询效率太低。

在关系模式的多种组合中选取一个好的关系模式的集合作为数据库模式，将会直接影响到整个数据库系统。那么，什么样的关系模式是相对较好的呢？人们通常依据规范化理论进行判断。

4.1.2 函数依赖

数据依赖是一个关系内部属性与属性之间的一种约束关系，这种约束关系是通过属性间值的相等与否体现出来的数据间的相互关系。数据依赖有多种类型，常用的数据依赖有函数依赖和多值依赖，其中函数依赖是最重要也是最基本的一种数据依赖。

1. 函数依赖的定义

函数依赖普遍地存在于现实生活中，它反映属性或属性组合之间相互依存、相互制约的关系。

函数依赖的定义为：设 $R(U)$是属性集 U 上的关系模式，X 与 Y 是 U 的子集，r 是 $R(U)$的任意一个可能的关系（即一个二维表）。如果对于 r 中的任意两个元组（即两个记录，或两行数据）t 和 s，由 $t[X]=s[X]$导致 $t[Y]=s[Y]$，则称 X 函数决定 Y，或称 Y 函数依赖于 X，记作 $X\rightarrow Y$。

函数依赖的相关术语和记号如下：

若 $X\rightarrow Y$，则称 X 为决定因素。

若 $X\rightarrow Y$，$Y\rightarrow X$，则记作 $X\longleftrightarrow Y$。

函数依赖是语义范畴的概念，需要根据语义来确定一个函数依赖。例如，在学生的关系

模式中“学生姓名→所在系别”这个函数依赖只有在学生没有重名的条件下才成立。如果允许有相同的学生姓名存在，则所在系别就不再函数依赖于学生姓名了。

2. 函数依赖的分类

关系数据库中函数依赖主要有以下几类。

1）平凡函数依赖和非平凡函数依赖

设 $R(U)$ 是属性集 U 上的关系模式，若对于任何 X、$Y \in U$，有 $X \rightarrow Y$ 且 Y 不包含于 X，则称 $X \rightarrow Y$ 是非平凡的函数依赖。反之，如果 Y 包含于 X，则称 $X \rightarrow Y$ 是平凡的函数依赖。

例如，在学生关系模式(学号，姓名，年龄，性别，所在系别)中，

学号→性别，(学号，姓名)→年龄，均为非平凡函数依赖。

(学号，姓名)→姓名，为平凡函数依赖。

若不特别声明，一般总是讨论非平凡的函数依赖。

2）完全函数依赖和部分函数依赖

设 $R(U)$ 是属性集 U 上的关系模式，如果 $X \rightarrow Y$，并且对于 X 的任何一个真子集 X'，都不存在 $X' \rightarrow Y$，则称 $X \rightarrow Y$ 是一个完全函数依赖，即 Y 完全函数依赖于 X，记作 $X \rightarrow Y^F$。

反之，如果存在 $X' \rightarrow Y$ 成立，则称 $X \rightarrow Y$ 是一个部分函数依赖，即 Y 部分函数依赖于 X，记作 $X \xrightarrow{P} Y$。

例如，在学生关系模式(学号，姓名，年龄，性别，所在系别)中，(学号，姓名)→年龄，为部分函数依赖。因为(学号，姓名)属性组合中存在真子集学号，使得“学号→年龄”也成立，所以它是部分函数依赖。学号→年龄，为完全函数依赖。

在选课关系模式(学号，课程号，成绩)中，(学号，课程号)→成绩，为完全函数依赖。

3）传递函数依赖

设 $R(U)$ 是属性集 U 上的关系模式，如果 $X \rightarrow Y$，$Y \rightarrow Z$，并且不存在 $Y \rightarrow X$，则称 $X \rightarrow Z$ 是一个传递函数依赖，即 Z 传递函数依赖于 X。

例如，在职工关系模式(职工编号，姓名，所在车间，车间主任)中，职工编号→所在车间，所在车间→车间主任，并且不存在所在车间→职工编号，则车间主任传递函数依赖于职工编号。

注意上述定义中的条件不存在 $Y \rightarrow X$。如果不加上这一限制，当 $X \rightarrow Y$ 时允许 $Y \rightarrow X$，则 $X \longleftrightarrow Y$。而在 $X \longleftrightarrow Y$ 的条件下，$Y \rightarrow Z$ 就等于 $X \rightarrow Z$。这样 X 就直接函数决定 Z，而不是通过 Y 传递决定 Z 了，即非传递函数依赖。

4.1.3 范式

在关系数据库中，关系模式设计的好坏取决于它的函数依赖是否满足特定的要求。满足特定要求的模式称为范式，满足不同程度要求的为不同范式。

1971—1972 年，E. F. Codd 首先提出了规范化理论，系统地提出了第一范式(简称

1NF)、第二范式(简称 2NF)和第三范式(简称 3NF)的概念。1974 年,E. F. Codd 和 Boyce 又共同提出了一个新的范式,即 BCNF(Boyce-Codd Normal Form,修正的第三范式)。1976 年,Fagin 提出了第四范式,后来又有人提出了第五范式。

一般地,关系模式 R 为第几范式就可以写成 $R \in x$NF。对于各种范式之间的联系有 5NF⊂4NF⊂BCNF⊂3NF⊂2NF⊂1NF 成立。

通过关系模式分解,可以将一个低一级范式的关系模式转换为若干个高一级范式的关系模式的集合,这种过程就叫规范化。

1. 第一范式

如果关系模式 R 中的每一个属性都是不可分解的,则称 R 属于第一范式,记作 $R \in$ 1NF。

例如,设关系模式 R(系别名称,高级职称人数)表示某学校系别的基本信息,假设系别信息状况如表 4.1 所示。

表 4.1　系别基本信息表

系别名称	高级职称人数	
	教授人数	副教授人数
侦查系	7	12
刑事技术系	5	10
公安信息系	5	8

从表 4.1 中可以看出,“高级职称人数”属性是可以分解的,所以 R 不满足 1NF。

解决问题的办法是:将“高级职称人数”属性拆开,形成关系模式 R_1(系别名称、教授人数、副教授人数)。形式如表 4.2 所示。显然,此时关系模式 R_1 中的每一个属性列都是不可再分的,所以 $R_1 \in$ 1NF。

表 4.2　分解后的系别基本信息表

系别名称	教授人数	副教授人数
侦查系	7	12
刑事技术系	5	10
公安信息系	5	8

第一范式是对关系模式最起码的要求。不满足第一范式的数据库模式不能称为关系数据库,但是满足第一范式的关系模式并不一定是一个好的关系模式。

2. 第二范式

如果关系模式 $R \in$ 1NF,且每一个非主属性都完全函数依赖于候选码,则称 R 属于第二范式,记作 $R \in$ 2NF。

例如,设关系模式 R(仓库号,设备号,数量,地点)表示仓库设备的存储情况。候选码是

(仓库号,设备号)属性组合,由于关系模式 R 中的每一个属性都不可再分,所以 $R \in 1NF$。因为非主属性“数量”完全函数依赖于候选码。非主属性“地点”部分函数依赖于候选码。即有(仓库号,设备号)→地点,仓库号→地点,所以 R 不满足 2NF。

关系模式 R 中存在异常,比如某一个仓库只有一种设备,当这种设备被移走后,在删除此设备信息的同时将这个仓库的信息也删除了。

解决问题的办法是:用投影分解把关系模式 R 分解为两个关系模式。将部分函数依赖关系的决定方属性和非主属性从关系模式中提出,单独构成一个关系模式;将余下属性加上码(仍要保留部分函数依赖的决定方属性)构成另一关系模式。

按照上述方法分解,将关系模式 R 分解为 $R1$(仓库号,设备号,数量)和 $R2$(仓库号,地点)两个关系模式。此时,$R1$ 和 $R2$ 均属于第二范式。

3. 第三范式

如果关系模式 $R \in 2NF$,且每一个非主属性都不传递函数依赖于候选码,则称 R 属于第三范式,记作 $R \in 3NF$。

例如,设关系模式 R(仓库号,仓库面积,所在城市,所在省)表示不同仓库在各省市分布情况。候选码是仓库号,由于关系模式 R 中的每一个属性都不可再分,所以 $R \in 1NF$。又因为 R 中每一个非主属性都完全函数依赖于候选码,所以 $R \in 2NF$。又因为函数依赖有仓库号→所在城市,所在城市→所在省,所以仓库号→所在省,R 中存在传递函数依赖,所以 R 不满足 3NF。

关系模式 R 中存在异常,比如要在辽宁省大连市设立一个仓库,此时想先存入有关所在城市的信息,但由于没有仓库号,主码为空,则插入是失败的。

解决问题的办法是:用投影分解把关系模式 R 分解为两个关系模式,将传递函数依赖的属性分解出来,消除传递函数依赖。

按照上述方法分解,将关系模式 R 分解为 $R1$(仓库号,仓库面积,所在城市)和 $R2$(所在城市,所在省)两个关系模式。此时,$R1$ 和 $R2$ 均属于第三范式。

4. BCNF(修正的第三范式)

如果关系模式 $R \in 3NF$,且没有一个属性部分函数依赖或传递函数依赖于候选码,则称 R 属于修正的第三范式,记作 $R \in BCNF$。

规范化的基本思想是,逐步消除数据依赖中不合理的部分,使每一个关系模式更趋于完美。但并不是范式越高越好,范式越高,模式分解的越多,在进行数据查询的时候往往要进行许多张表的连接,系统开销较大,查询效率较低。所以,在进行关系模式规范化的过程中,关系模式一般分解到 3NF 就认为是比较好的了。

例题 4.1 在银行管理系统的数据库中,有一关系模式为 R(BNO,SSNO,BNAME,ADDRESS,CITY,SNAME,SEX,AGE,ACCOUNT),其中属性分别表示银行编号,身份证号,银行名称,银行所在地点,银行所在城市,顾客姓名,性别,年龄,账户号。写出该关系模式的主码,并判断此关系模式是否满足 3NF,若不满足请对其进行规范化,以达到 3NF。

例题解析：

该关系模式 R 的主码为(BNO,SSNO)。由于关系模式 R 中的每个分量都是不可再分的数据项，所以 R 满足 1NF。关系模式 R 中存在以下函数依赖：

(BNO,SSNO)→BNAME,　　BNO→BNAME,
(BNO,SSNO)→ADDRESS,　　BNO→ADDRESS,
(BNO,SSNO)→CITY,　　ADDRESS→CITY,
(BNO,SSNO)→ACCOUNT,　　BNO→CITY,
(BNO,SSNO)→SNAME,　　SSNO→SNAME,
(BNO,SSNO)→SEX,　　SSNO→SEX,
(BNO,SSNO)→AGE,　　SSNO→AGE,

首先，关系模式 R 满足 1NF，但存在部分函数依赖，所以，R 不满足 2NF，将其分解为：

$R1$(BNO,SSNO,ACCOUNT)∈2NF；

$R2$(BNO,BNAME,ADDRESS,CITY)∈2NF；

$R3$(SSNO,SNAME,SEX,AGE)∈2NF；

其次，关系模式 $R1$、$R3$ 均已满足第三范式，但关系模式 $R2$ 存在传递函数依赖，$R2$ 不满足第三范式，将 $R2$ 分解为：

$R4$(BNO,BNAME,ADDRESS)∈3NF；

$R5$(ADDRESS,CITY)∈3NF；

最后，$R1$、$R3$、$R4$、$R5$ 满足第三范式，总结为：

$R1$(BNO,SSNO,ACCOUNT)；

$R3$(SSNO,SNAME,SEX,AGE)；

$R4$(BNO,BNAME,ADDRESS)；

$R5$(ADDRESS,CITY)。

4.2　数据库设计概述

什么是数据库设计呢？具体地说，数据库设计是要在一个给定的应用环境中，通过合理的逻辑设计和有效的物理设计，构造较优的数据库模式，建立数据库及其应用系统，能够有效地存储和管理数据，满足用户的各种信息需求。因此，数据库设计是数据库在应用领域的主要研究课题。

4.2.1　数据库设计的方法

采用合理的数据库设计方法，可以确保数据库系统的设计质量，降低系统运行后的维护代价。数据库设计是涉及多学科的综合性技术，也是一项庞大的软件开发工程。因此，一个从事数据库设计的专业人员应该具备多方面的专业技术和知识。除了具备计算机科学的基

础知识之外，还必须了解软件工程的原理，掌握程序设计的技巧；具备数据库的基本知识和数据库设计技术，同时还必须具备应用领域的专业知识，才能设计出符合具体应用领域要求的数据库应用系统。

早期数据库设计主要采用手工与经验相结合的方法。设计的质量往往与设计人员的经验与水平有直接的关系，设计质量难以保证。经常是数据库运行一段时间后又出现各种各样不同的问题，需要进行修改或重新设计，大大增加了后期维护的负担。所以人们努力探索，通过运用软件工程的思想和方法，提出了各种数据库设计方法，以及各种设计准则和规程，这些都属于规范设计方法。例如：

(1) 关系模式的设计方法。

(2) 新奥尔良(New Orleans)方法。

(3) 基于 E-R 模型的数据库设计方法。

(4) 3NF(第三范式)的设计方法。

(5) 基于抽象语法规范的设计方法。

(6) 计算机辅助数据库设计方法。

这些数据库设计方法中比较著名的是新奥尔良(New Orleans)方法，它将数据库设计分为四个阶段：需求分析(分析用户要求)、概念设计(信息分析和定义)、逻辑设计(设计实现)和物理设计(物理数据库设计)。

4.2.2 数据库设计的步骤

从数据库应用系统设计和开发的全过程来考虑，一般将数据库设计的步骤分为七个阶段：系统规划阶段、需求分析阶段、概念结构设计、逻辑结构设计、物理结构设计、实施阶段、运行和维护阶段。

1. 系统规划阶段

系统规划阶段是确定数据库系统在整个企业管理系统中的地位，确定系统的范围，确定开发工作所需的资源(人员、硬件和软件)，确定项目进度，估算软件开发的成本及系统可能达到的效益。

2. 需求分析阶段

需求分析阶段是整个设计过程的基础，是最困难、最耗费时间的一个阶段。这一阶段要求计算机人员(系统分析员)和用户共同收集数据库所需要的信息内容，以及用户对处理的要求，并加以规格化和分析，以书面形式确定下来，作为以后验证系统的依据。

3. 概念结构设计阶段

概念结构设计阶段是整个数据库设计的关键。它通过对用户需求进行综合、归纳与抽象，形成一个独立于具体 DBMS 的概念模型，可以用 E-R 图来表示。

4. 逻辑结构设计阶段

逻辑结构设计阶段是将概念结构(E-R 图)转换为某个 DBMS 所支持的数据模型，并对

其进行优化。

5. 物理结构设计阶段

数据库的物理结构设计阶段是为逻辑数据模型选取一个最适合应用环境的物理结构（包括存储结构和存取方法）。

6. 实施阶段

数据库设计人员运用 DBMS 提供的数据库语言及其宿主语言，根据逻辑设计和物理设计的结果建立数据库，编制与调试应用程序，组织数据入库，并进行试运行。

7. 运行和维护阶段

数据库应用系统经过试运行后即可投入正式运行。在数据库系统运行过程中必须不断地对其进行评价、调整与修改。

数据库设计的七个阶段的划分目前尚无统一的标准，各阶段间相互连接，而且常常需要回溯修正。

4.3　系统规划阶段

4.3.1　系统规划的任务

系统规划阶段的主要任务就是进行系统的必要性和可行性分析。包括明确应用系统的基本功能，划分数据库支持的范围；规划人力资源调配；拟定设备配置方案；选择合适的操作系统、DBMS 和其他软件；设备配置方案要在使用要求、系统性能、购置成本和维护代价各方面综合权衡；对系统的开发、运行、维护的成本做出估算；预测系统效益的期望值；拟定开发进度计划，还要对现行工作模式如何向新系统过渡作出具体安排。

4.3.2　系统规划的成果

规划阶段的工作成果是写出详尽的可行性分析报告和数据库应用系统规划书。内容应包括：系统的定位及其功能、数据资源及数据处理能力、人力资源调配、设备配置方案、开发成本估算、开发进度计划等。

可行性分析报告和数据库应用系统规划书经审定立项后，成为后续开发工作的总纲。

4.4　需求分析阶段

4.4.1　需求分析的任务

需求分析是整个数据库设计过程中最重要的步骤之一，是后续各阶段的基础。需求分析的主要任务是通过详细调查所要处理的对象，包括某个组织、某个部门、某个企业的业务管理等，充分了解原手工或原计算机系统的工作状况以及工作流程，明确用户的各种需求，

生成业务流程图和数据流图，然后在此基础上确定新系统的功能，并撰写系统说明书。新系统不能只按当前应用需求来设计数据库，必须充分考虑今后可能的扩充和改变。

在需求分析阶段，从多方面对整个组织进行调查、收集和分析各项应用对信息和处理两方面的需求。需求分析的重点是调查、收集和分析用户对数据管理中的信息要求、处理要求、安全性与完整性要求。信息要求是指用户需要从数据库中获得信息的内容与性质。由信息要求可以导出数据要求，即在数据库中需要存储哪些数据。处理要求是指用户要完成什么处理功能，对处理的响应时间有什么要求，处理方式是批处理还是联机处理。新系统的功能必须能够满足用户的多种需求。

4.4.2 需求分析的步骤

调查、收集和分析用户要求的具体步骤如下：

1. 调查组织机构情况

调查这个组织由哪些部门组成，各部门担当的职责是什么。

2. 调查各部门的业务活动情况

调查各部门所需输入和使用的数据，如何加工处理这些数据，输出什么信息，输出到哪个部门，输出结果的格式等。

3. 协助用户明确对新系统的各种要求

进一步明确用户对数据管理中的信息要求、处理要求、安全性与完整性要求。

4. 确定新系统的边界

确定哪些功能由计算机完成或将来准备让计算机完成，哪些功能由人工完成。由计算机完成的功能就是新系统应该实现的功能。

4.4.3 需求分析的调查方法

根据不同的问题和条件，调查方法也可以不同。常用的调查方法有以下几种。

1. 跟班作业

通过亲身参加业务工作来了解业务活动的情况，这种方法可以比较准确地了解用户的需求，但比较耗费时间。

2. 开调查会

通过与用户座谈的方式来了解业务活动情况及用户需求。

3. 请专人介绍

通过邀请熟悉业务的专业人士来了解业务活动情况。

4. 询问

对调查中的某些问题，可以找专人询问。

5. 设计调查表请用户填写

如果调查表设计合理，这种方法易于用户接受并且会很有效。

6. 查阅记录

查阅与原系统有关的数据记录，包括原始的单据、报表等。

当需求分析完成后，最终产生阶段性的成果：系统需求说明书，包括数据流图、数据字典、数据表格、系统功能结构图和必要的说明。

4.4.4 数据流图

数据流图(Data Flow Diagram，DFD)是用图形方式来表达系统的逻辑功能，以及数据在系统内部的逻辑流向和逻辑变换过程。任何一个系统都可以抽象为图 4.1 所示的数据流图形式。

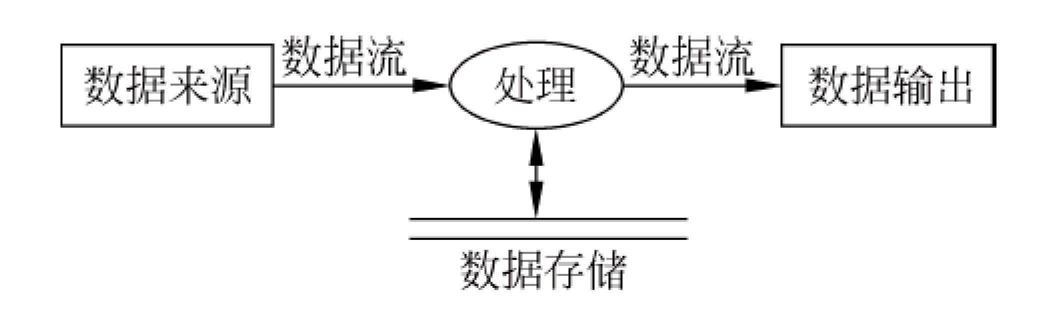


图 4.1　数据流图

1. 数据流图的基本符号

→：箭头，表示数据流。

□：方框，表示数据的源点或终点。

○：圆或椭圆，表示加工或处理。

＝：双杠，表示数据存储。

(1) 数据流：是数据在系统内传播的路径，因此由一组成分固定的数据组成。例如订票单由旅客姓名、年龄、单位、身份证号、日期、目的地等数据项组成。由于数据流是流动中的数据，所以必须有流向，除了与数据存储之间的数据流不用命名外，数据流应该用名词或名词短语命名。

(2) 数据源点或终点：代表系统之外的实体，可以是人、物或其他软件系统。

(3) 对数据的加工(处理)：是对数据进行处理的单元，它接收一定的数据输入，对其进行处理，并产生输出。

(4) 数据存储：表示信息的静态存储，可以代表文件、文件的一部分、数据库的元素等。

2. 在画数据流图时须注意的原则

(1) 一个加工的输出数据流不应与输入数据流同名，即使它们的组成成分相同。

(2) 保持数据守恒，即一个加工的所有输出数据流中的数据必须能从该加工的输入数据流中直接获得，或者说是通过该加工能产生的数据。

(3) 每个加工必须既有输入数据流，又有输出数据流。

(4) 所有的数据流必须以一个加工开始，或以一个加工结束。

3. 数据流图的实例

如图 4.2 是一个飞机机票预订系统的数据流图，它反映的功能是：旅行社把预订机票

的旅客信息(姓名、年龄、性别、身份证号码、旅行时间、目的地等)输入机票预订系统。系统为旅客安排航班,打印出取票通知单(附有应交的账款)。旅客在飞机起飞的前一天凭取票通知单交款取票,系统检验无误,输出机票给旅客。

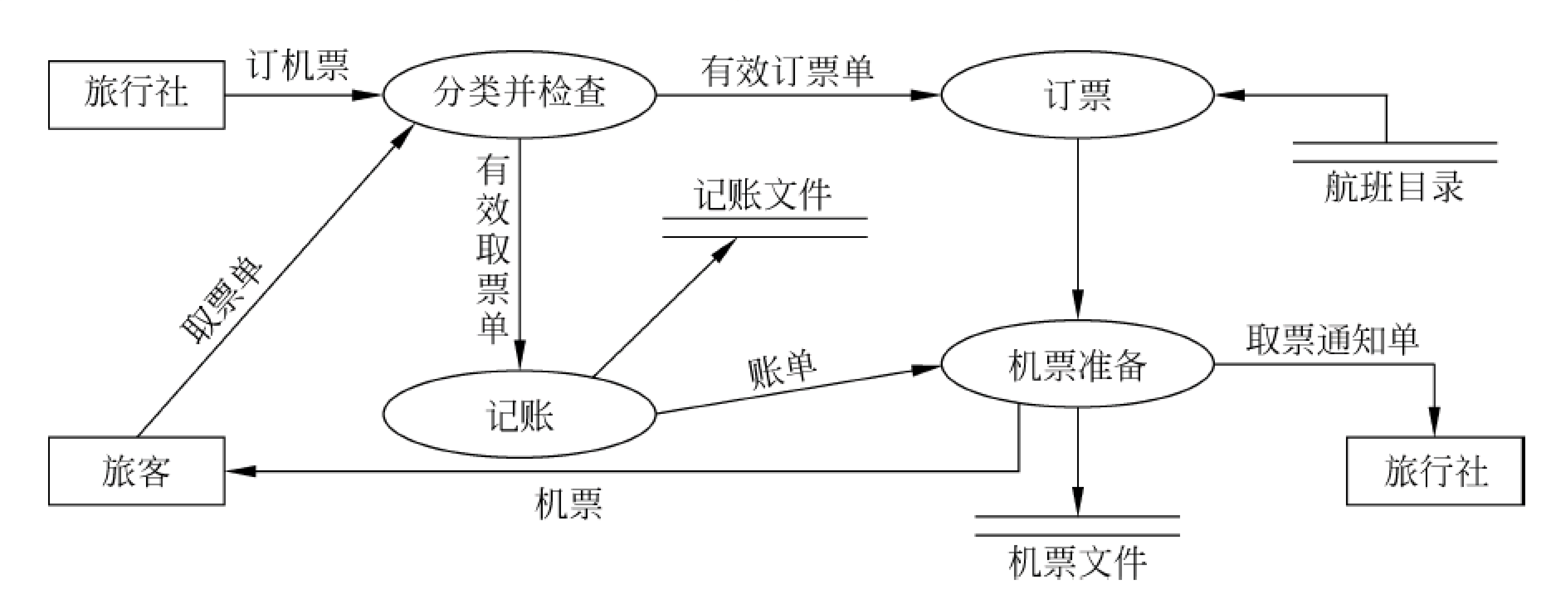


图 4.2 飞机机票预订系统

4.4.5 数据字典

数据字典是系统中各类数据描述的集合,是对数据流图中包含的所有元素的定义的集合。

数据存放于物理数据库中,由数据库管理系统进行管理。数据字典有助于对这些数据进一步管理和控制,为设计人员和数据库管理员在数据库设计、实现和运行阶段控制有关数据提供一定的依据。

数据字典通常包括数据项、数据结构、数据流、数据存储和处理过程五个部分。

(1) 数据项是数据的最小组成单位,是不可再分的数据单位。包括项名、含义说明、别名、数据类型、长度、取值范围、与其他数据项的逻辑关系等。

描述:

数据项名:学号。

数据项含义:唯一标识每个学生。

别名:学生编号。

类型:字符型。

长度:10。

取值范围:0000000000～9999999999。

取值含义:前四位标识该学生所在的年级,后六位按顺序编号。

与其他数据项的逻辑关系:该项等于另两项之和。

数据项之间的联系:根据语义写出数据项之间的数据依赖。

(2) 数据结构反映了数据之间的组合关系。一个数据结构可以由若干个数据项组成,也可以由若干个数据结构组成,或由若干个数据项和数据结构混合组成。包括数据结构名、说明、组成等。

描述：

数据结构名：学生。

含义说明：是学籍管理子系统的主体数据结构，定义了一个学生的相关信息。

组成：学号，姓名，年龄，性别，所在系，年级。

(3) 数据流是数据结构在系统内传输的路径。包括数据流名、说明、数据流来源、数据流去向、组成、平均流量、高峰期流量等。

描述：

数据流名：体检结果

说明：学生参加体格检查的最终结果

数据流来源：体检(说明该数据流来自哪个过程)

数据流去向：批准(说明该数据流将到哪个过程去)

组成：身高，体重，视力，血压……

平均流量：单位时间内传输的次数

高峰期流量：最高时期的数据流量

(4) 数据存储说明数据流中需要存储的数据，包括数据存储名、说明、流入数据流、流出数据流、组成、数据量、存取频度、存取方式等。

描述：

数据存储名：学生登记表

说明：记录学生的基本信息

输入数据流：指数据来源，如：报到时填的表

输出数据流：指数据去向，如：学生基本情况表

组成：数据结构或数据项，如：学号，姓名，年龄，性别，所在系，年级，专业等

数据量：如每年 5000 张

存取频度：指每小时或每天或每周存取几次、每次存取多少数据等信息

存取方式：指是批处理，还是联机处理；是检索，还是更新；是顺序检索，还是随机检索

(5) 处理过程的具体处理逻辑通常用判定表或判定树来描述。包括处理过程名、说明、输入数据流、输出数据流、处理简要说明等。

描述：

处理过程名：分配宿舍

说明：为所有新生分配宿舍

输入数据流：如学生、宿舍

输出数据流：宿舍安排

处理：在新生报到后，为所有新生分配宿舍，要求相同性别的学生才可以居住在同一个房间里，一个人只能有一间宿舍，每个人的居住面积大于等于 3 平方米，安排新生宿舍的处理时间不得超过 20 分钟。

4.5 概念结构设计

4.5.1 概念结构设计方法

概念结构设计是整个数据库设计的关键,其主要任务是在需求分析阶段产生的需求说明书的基础上,按照特定的方法把它们抽象为一个不依赖于任何具体机器的数据模型,即概念模型。

概念结构的设计方法通常有以下四种。

(1) 自顶向下:先定义全局概念结构 E-R 模型的框架,再逐步细化。

(2) 自底向上:先定义各局部应用的概念结构 E-R 模型,然后将它们集成,得到全局概念结构 E-R 模型。

(3) 逐步扩张:先定义最重要的核心概念 E-R 模型,然后向外扩充,以滚雪球的方式逐步生成其他概念结构 E-R 模型,直至总体概念结构。

(4) 混合策略:该方法采用自顶向下和自底向上相结合的方法,先自顶向下定义全局框架,再以它为骨架集成自底向上方法中设计的各个局部概念结构。

其中最经常采用的策略是自底向上方法,即自顶向下地进行需求分析,然后再自底向上地设计概念结构。主要步骤包括进行数据抽象,设计局部概念结构,将局部概念结构合并成全局概念结构,并进行优化。

4.5.2 E-R 设计方法的介绍

描述概念模型的有力工具是 E-R 模型。有关 E-R 模型的基本概念已经在第 1 章介绍过了,下面将用 E-R 模型来描述概念结构。

1. E-R 方法的基本术语

E-R 方法是"实体-联系方法"(Entity-Relationship Approach)的简称。它是描述现实世界概念结构模型的有效方法。用 E-R 方法建立的概念结构模型称为 E-R 模型,或称为 E-R 图。

E-R 图的三要素是实体、属性和联系。

(1) 实体:用矩形框表示,框内标注实体名称。如图 4.3 所示。

学生　课程　教师

图 4.3 实体

(2) 属性:用椭圆形框表示,框内标注属性名称。如图 4.4 所示。

学号　姓名　课程号　课程名　教师号　成绩

图 4.4 属性

(3) 联系：用菱形框表示，框内标注实体之间的关系。有 1∶1、1∶n 和 m∶n 三种联系类型。例如系主任领导系，学生选修课程，教师讲授课程，工人生产产品，这里"领导"、"选修"、"讲授"、"生产"表示实体之间的联系，可以作为联系名称。联系用菱形框表示，框内标注联系名称。如图 4.5 所示。

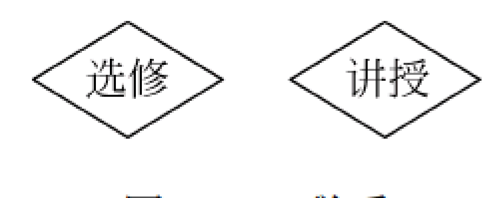


图 4.5　联系

2. E-R 图的表示

在 E-R 图的描述中，用矩形表示实体，用椭圆表示属性，用菱形表示联系。在各框图内标注它们的名称，它们之间用无向线连接，表示联系时需在线上标明属于哪种类型的联系。如图 4.6 所示。

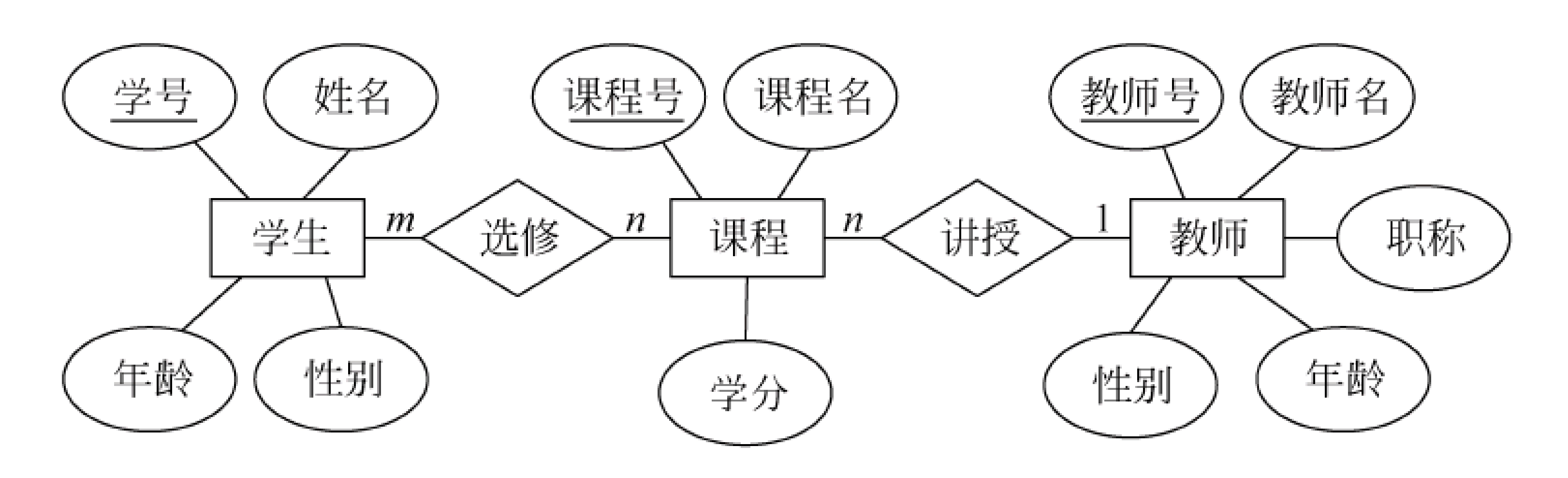


图 4.6　E-R 图的表示

采用 E-R 方法进行概念结构设计，可以按照局部概念结构设计阶段和全局概念结构设计阶段两步进行，在全局概念结构设计的过程中要不断进行概念结构的优化。

4.5.3　局部概念结构设计

概念结构设计首先要根据需求分析得到的结果(数据流图、数据字典等)对现实世界进行抽象，设计各个局部 E-R 模型。在系统需求分析阶段，最后得到了多层数据流图、数据字典和系统分析报告。建立局部 E-R 模型，就是根据系统的具体情况，在多层的数据流图中选择一个适当层次的数据流图，作为设计局部 E-R 图的出发点，让这组图中每一部分对应一个局部应用。在前面选好的某一层次的数据流图中，每个局部应用都对应了一组数据流图，局部应用所涉及的数据存储在数据字典中。现在就是要将这些数据从数据字典中抽取出来，参照数据流图，确定每个局部应用包含哪些实体，这些实体又包含哪些属性，以及实体之间的联系及其联系类型。局部 E-R 模型设计的步骤如图 4.7 所示。

例题 4.2　以工厂管理为例，描述局部 E-R 图的设计。从技术科获知，每种产品由多种零件组成，每种零件可用在不同的产品上，每种产品由一定数量的零件组成。从供应科获知，每种零件使用多种材料制成，每种材料也可应用在不同的零件上，每种零件在使用材料

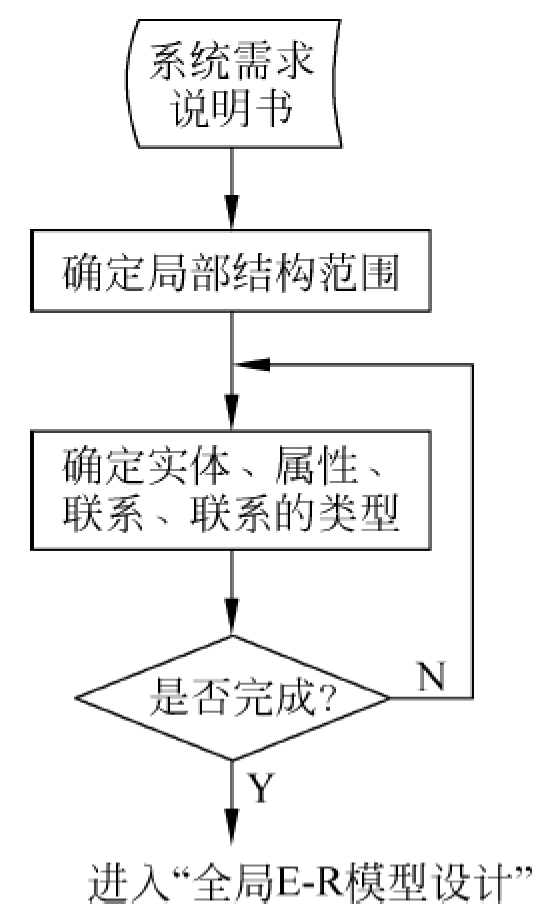


图 4.7　局部 E-R 模型设计的步骤

上有一个使用量；每个仓库可以存放多种材料，每种材料只能放在一个仓库里，每个仓库存放材料有一个库存量。

根据 E-R 图的建立过程：

第一步，确定实体类型。

产品、零件、材料和仓库四个实体类型。

第二步，确定联系类型。

产品和零件之间是 $m:n$ 组成的联系，零件和材料之间是 $m:n$ 使用的联系，仓库和材料之间是 $1:m$ 存放的联系。

第三步，确定实体类型和联系类型的属性。

在技术科中，产品实体的属性有产品号、产品名、性能参数等。

在技术科中，零件实体的属性有零件号、零件名、价格等。

在供应科中，零件实体的属性有零件号、规格等。

在供应科中，材料实体的属性有材料号、价格等。

在供应科中，仓库实体的属性有仓库号、仓库名、地址等。

产品和零件之间 $m:n$ 组成的联系属性是零件数，零件和材料之间 $m:n$ 使用的联系属性是使用量，仓库和材料之间 $1:m$ 存放的联系属性是库存量。

第四步，根据实体类型和联系类型画出局部 E-R 图，如图 4.8 和图 4.9 所示。

第五步，用下划线标注出实体标识符。

4.5.4　全局概念结构设计

全局概念结构设计的实质是把局部概念结构设计中所有的局部概念模型统一起来，形成一个完整的系统模型。全局 E-R 模型的设计过程如图 4.10 所示。

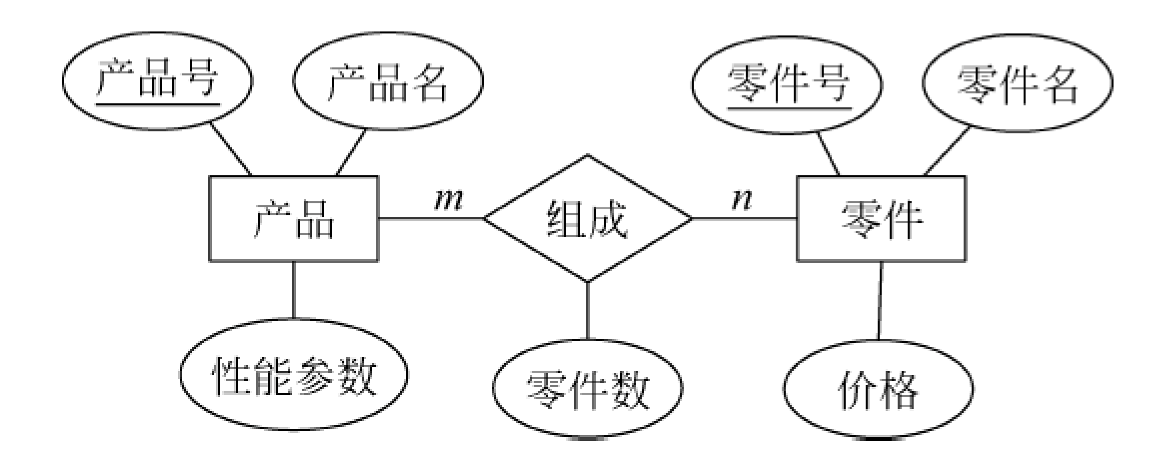


图 4.8　技术科的局部 E-R 图

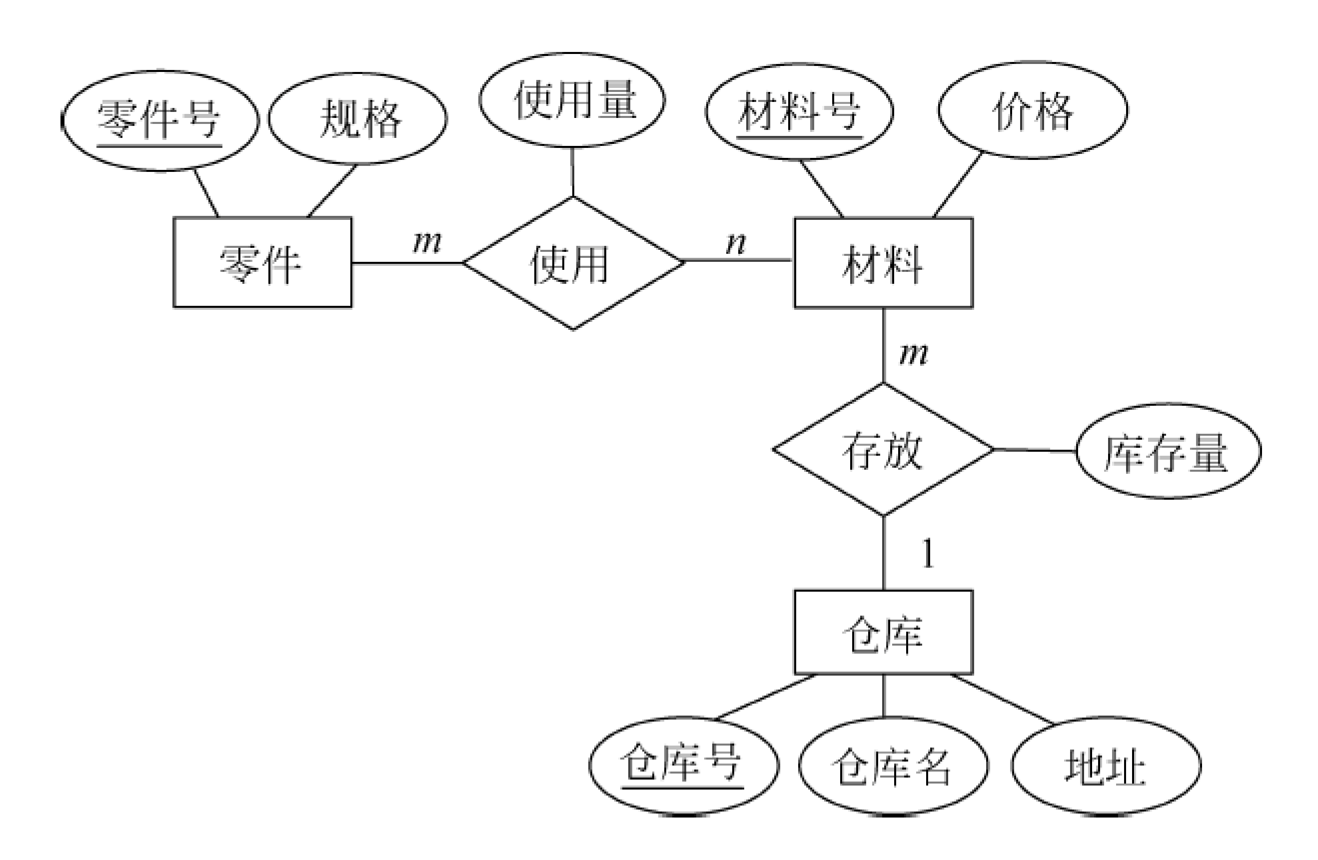


图 4.9　供应科的局部 E-R 图

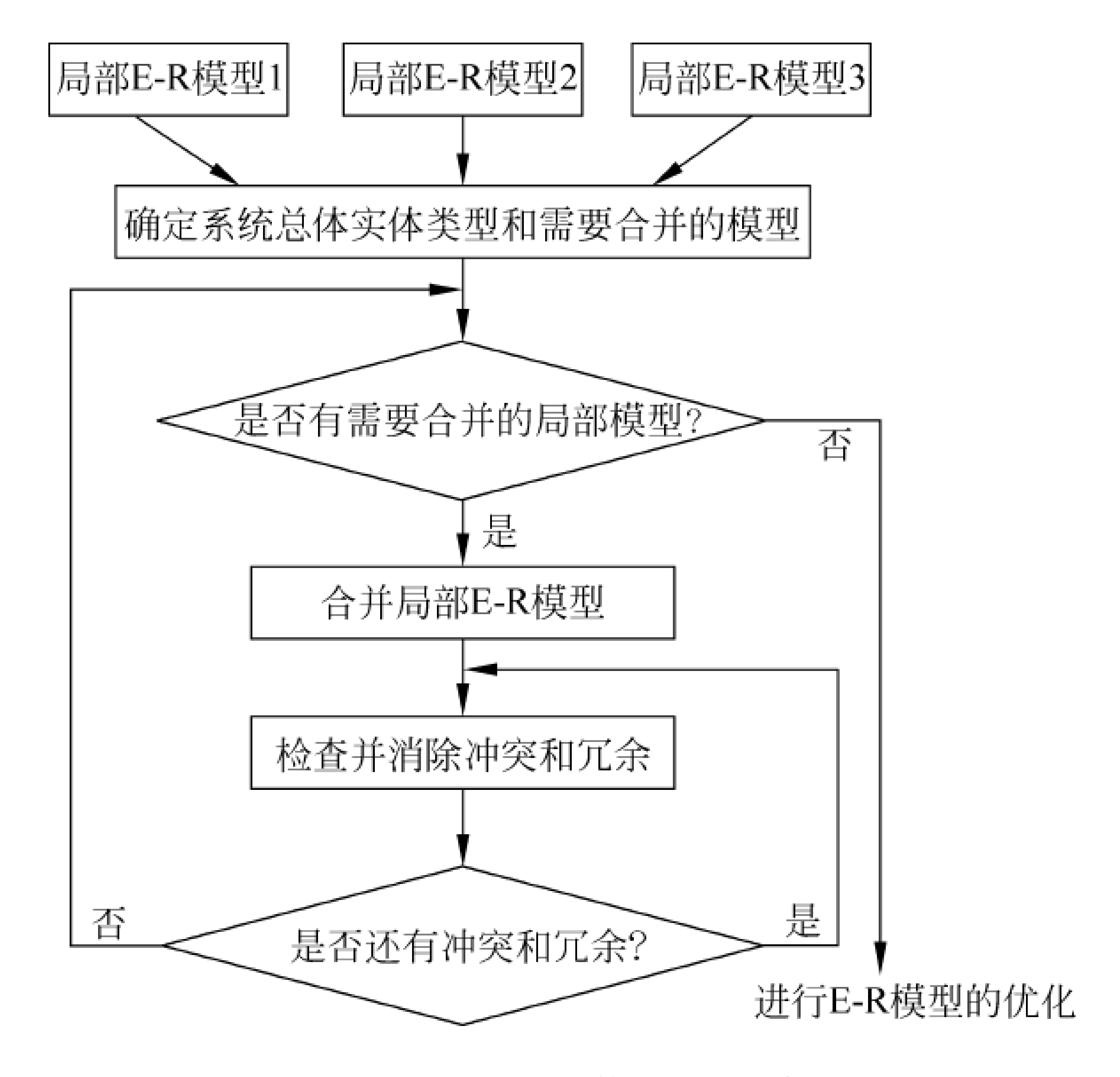


图 4.10　全局 E-R 模型的设计步骤

全局 E-R 模型的建立过程如下。

1. 合并

将局部概念模型整理合并成全局概念模型。

(1) 先找出具有相同实体的两个 E-R 图。

(2) 以该相同实体为基准进行合并。

(3) 如果还有相同实体的 E-R 图,再次合并。

(4) 这样一直下去,直到所有的具有相同实体的局部 E-R 图都被合并,从而得到全局的 E-R 图。

2. 消除冲突

解决各种局部 E-R 图之间的冲突问题,生成初步 E-R 图。

1) 属性冲突

属性值的类型、取值范围及取值单位不一致造成的冲突。如生日和年龄、厘米和米、学生编号的方式等。

2) 结构冲突

如在某局部 E-R 图中系主任是属性,而在另一个局部 E-R 图中系主任是实体等。

3) 命名(实体、属性、联系)冲突

同名异义:教室和宿舍均称为房间;

异名同义:如教材和课本。

将例题 4.2 中技术科和供应科的两个局部 E-R 图合并成全局 E-R 图。

图 4.11 是工厂管理中的两个局部 E-R 图按照相同的实体"零件"合并后得到的全局 E-R 图。

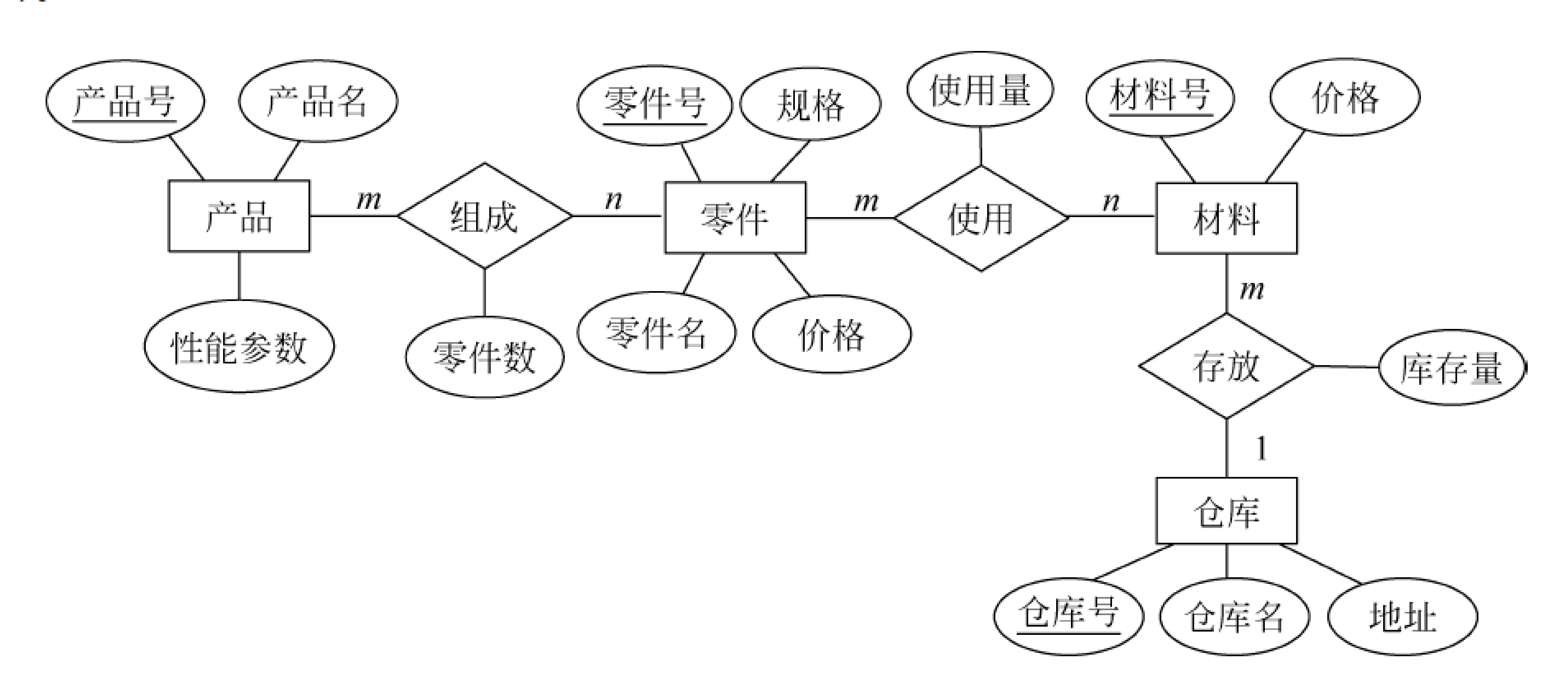


图 4.11 工厂管理的全局 E-R 图

按照上面的方法将各个局部 E-R 模型合并后就得到一个初步的全局 E-R 模型,之所以这样称呼是因为其中可能存在冗余的数据和冗余的联系等。因此,在得到初步的全局 E-R 模型后,还应当进一步检查 E-R 图中是否存在冗余,如果存在冗余则一般应设法将其消除。

一个好的全局 E-R 模型除了能准确、全面地反映用户功能外，还应满足下列条件：实体类型的个数尽可能少、实体类型所含属性的个数尽可能少、实体间联系的冗余最小。模型优化的目的是消除不必要的冗余，使其保持最小冗余度。

优化全局 E-R 模型的几个原则：

(1) 实体类型的合并，如图 4.12 中的"系主任"和"系"。

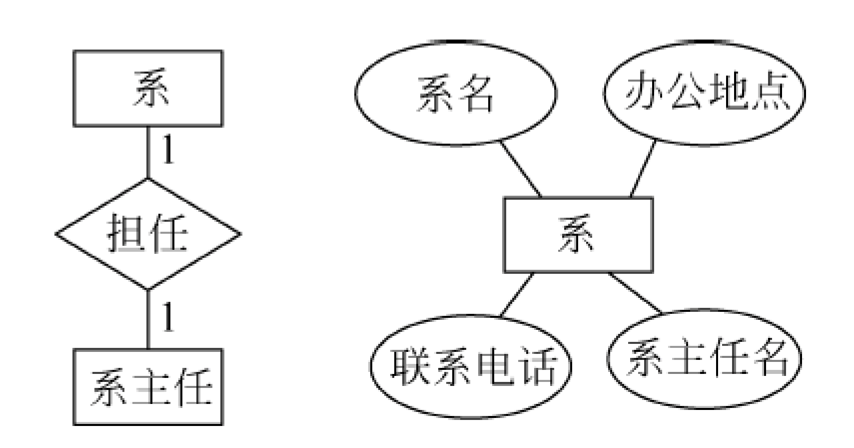


图 4.12　系主任和系实体类型的合并

(2) 冗余属性的消除，如生日和年龄。

(3) 冗余联系的消除。

将如图 4.13 和图 4.14 所示的两个局部 E-R 图合并成一个全局 E-R 图，并进行优化。

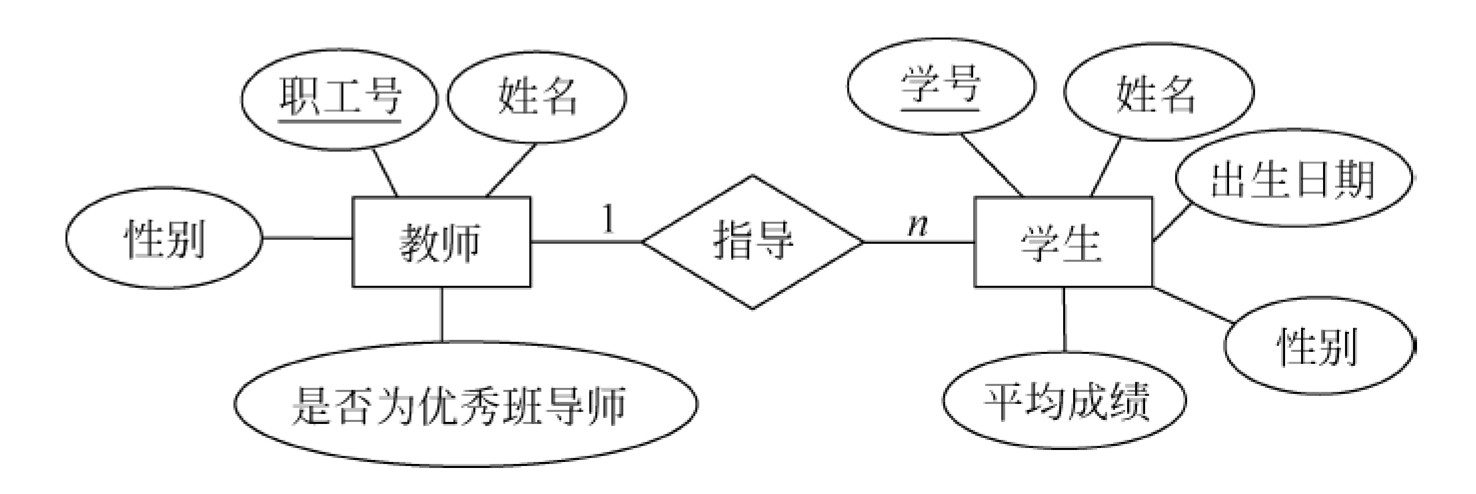


图 4.13　班导师工作局部 E-R 图

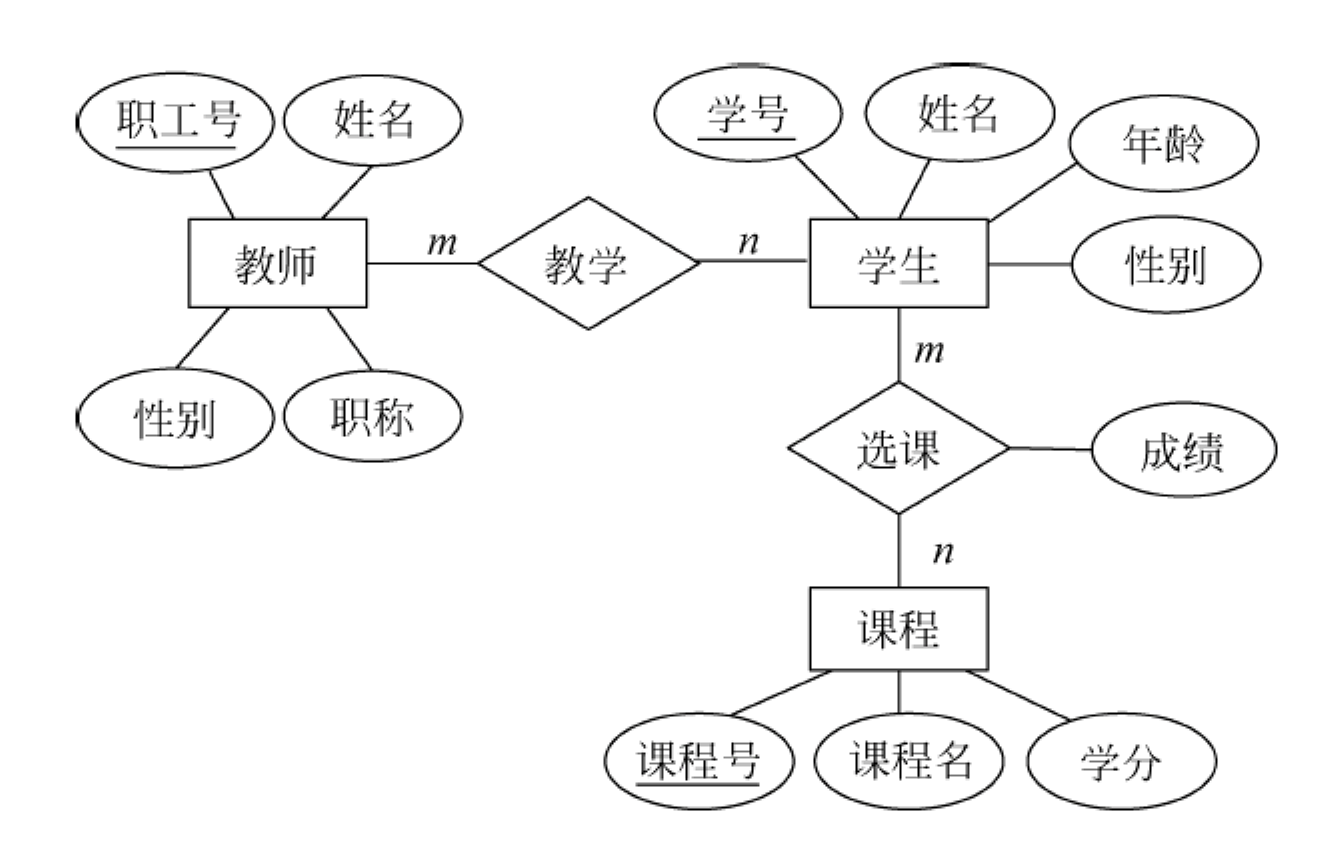


图 4.14　教学活动局部 E-R 图

局部 E-R 模型设计完成之后，根据全局 E-R 模型的建立步骤，将上述两个局部 E-R 模型合并成全局 E-R 模型如图 4.15 所示。

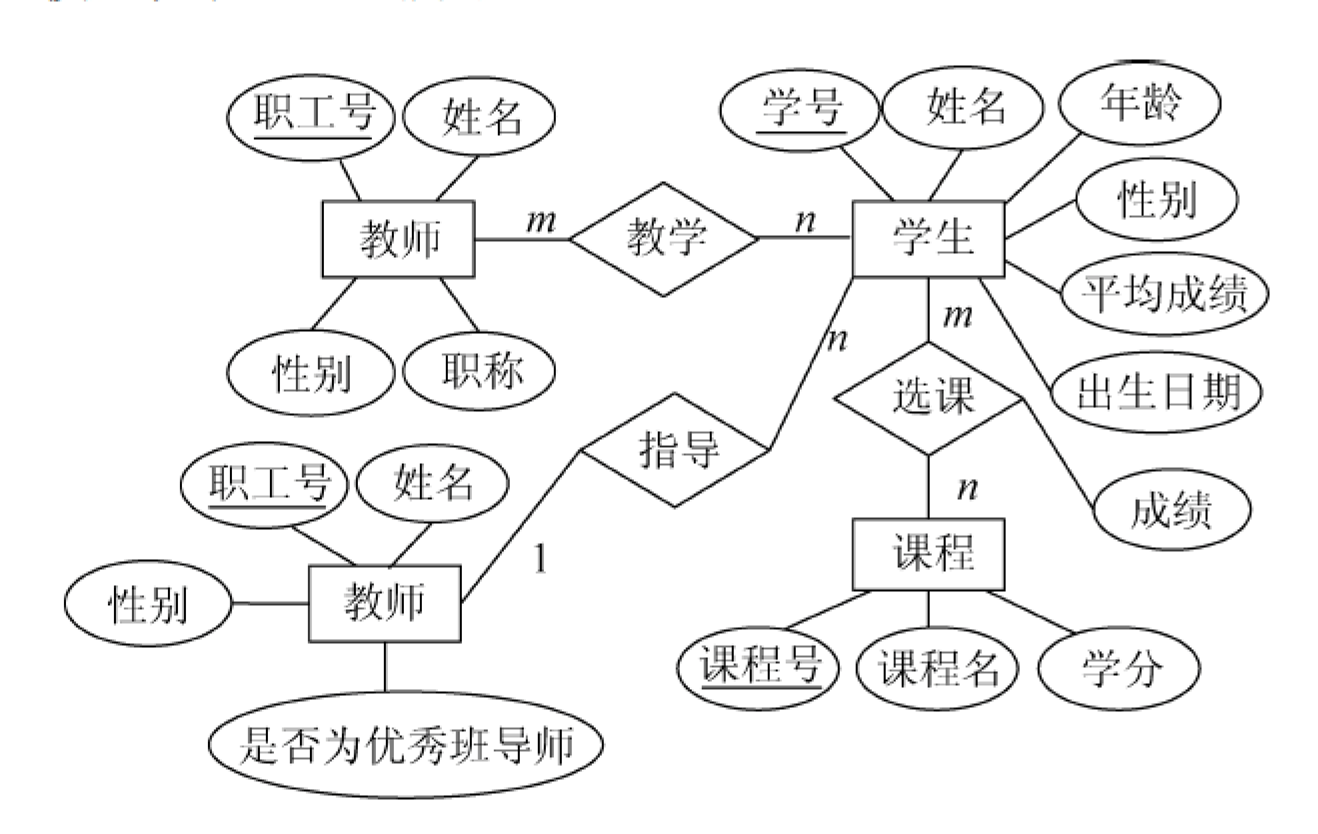


图 4.15　合并后的全局 E-R 图

全局概念结构不仅要支持所有的局部 E-R 模型，而且要合理地表示一个完整、一致的数据库概念结构。由于各个局部应用不同，通常由不同的设计人员进行局部 E-R 图设计，因此，各局部 E-R 图不可避免地会有许多不一致的地方，我们称之为冲突。

在图 4.15 合并的 E-R 模型中存在冲突。实体存在冗余，教学和指导两个联系存在冗余，年龄和出生日期两个属性存在冗余，成绩和平均成绩两个属性存在冗余。

合并局部 E-R 图时并不能简单地将各个局部 E-R 图画到一起，而必须消除各个局部 E-R 图中的不一致，使合并后的全局概念结构不仅支持所有的局部 E-R 模型，而且必须是一个能为全系统中所有用户共同理解和接受的完整的概念模型。上例中对全局 E-R 模型进行优化，消除冗余，得到优化后的概念模型如图 4.16 所示。

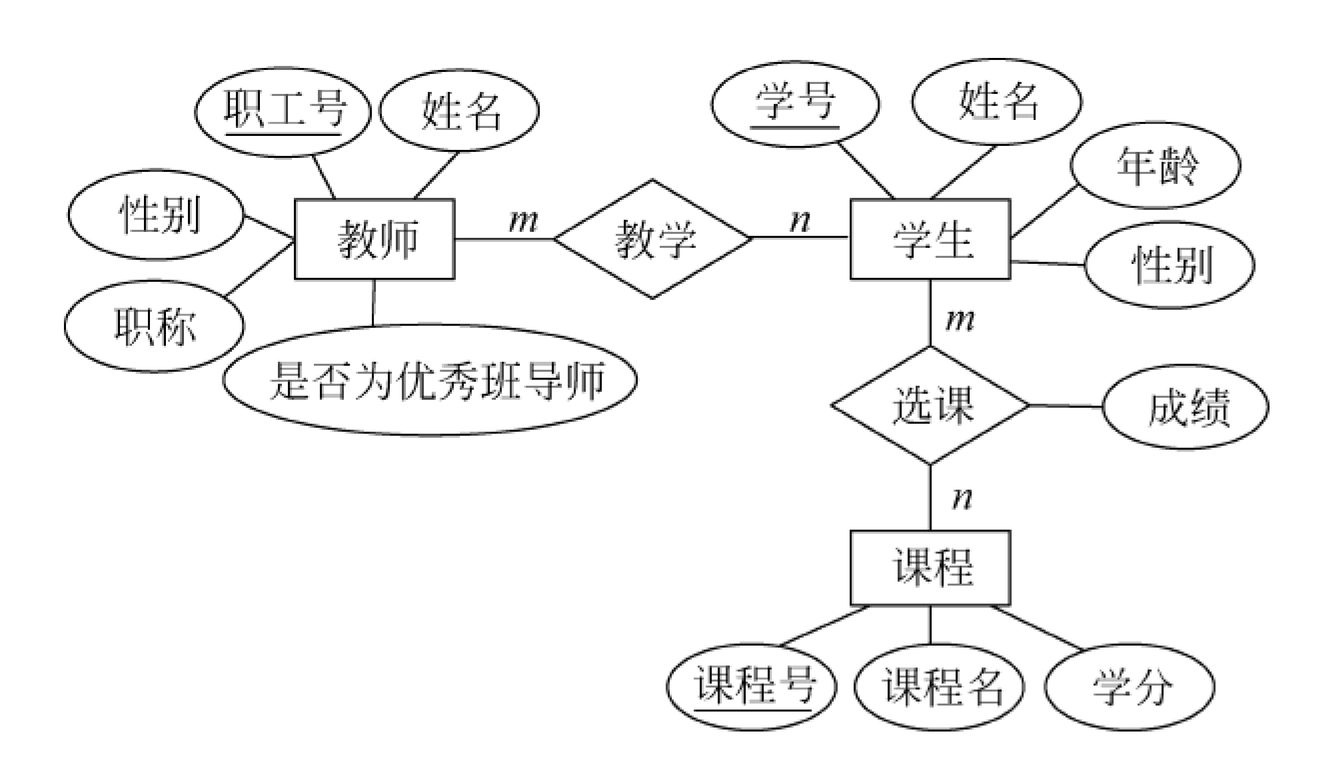


图 4.16　优化后的全局 E-R 图

例题 4.3　一个图书借阅信息管理系统有如下信息：

每一个借书人可以借阅多本图书，每一本图书可以被多个借书人借阅；借书人每借阅

一本图书都有一个借书日期和还书日期；每一个出版社可以出版多本图书，每一本图书只能在一个出版社出版。

其中，借书人的属性有借书证号、姓名、单位、电话；图书的属性有图书编号、书名、位置；出版社的属性有出版社名、地址、邮编、电话。

根据需求画出 E-R 图，并在 E-R 图中注明实体的属性、联系的类型以及实体的码。

图书借阅信息管理系统 E-R 图，如图 4.17 所示。

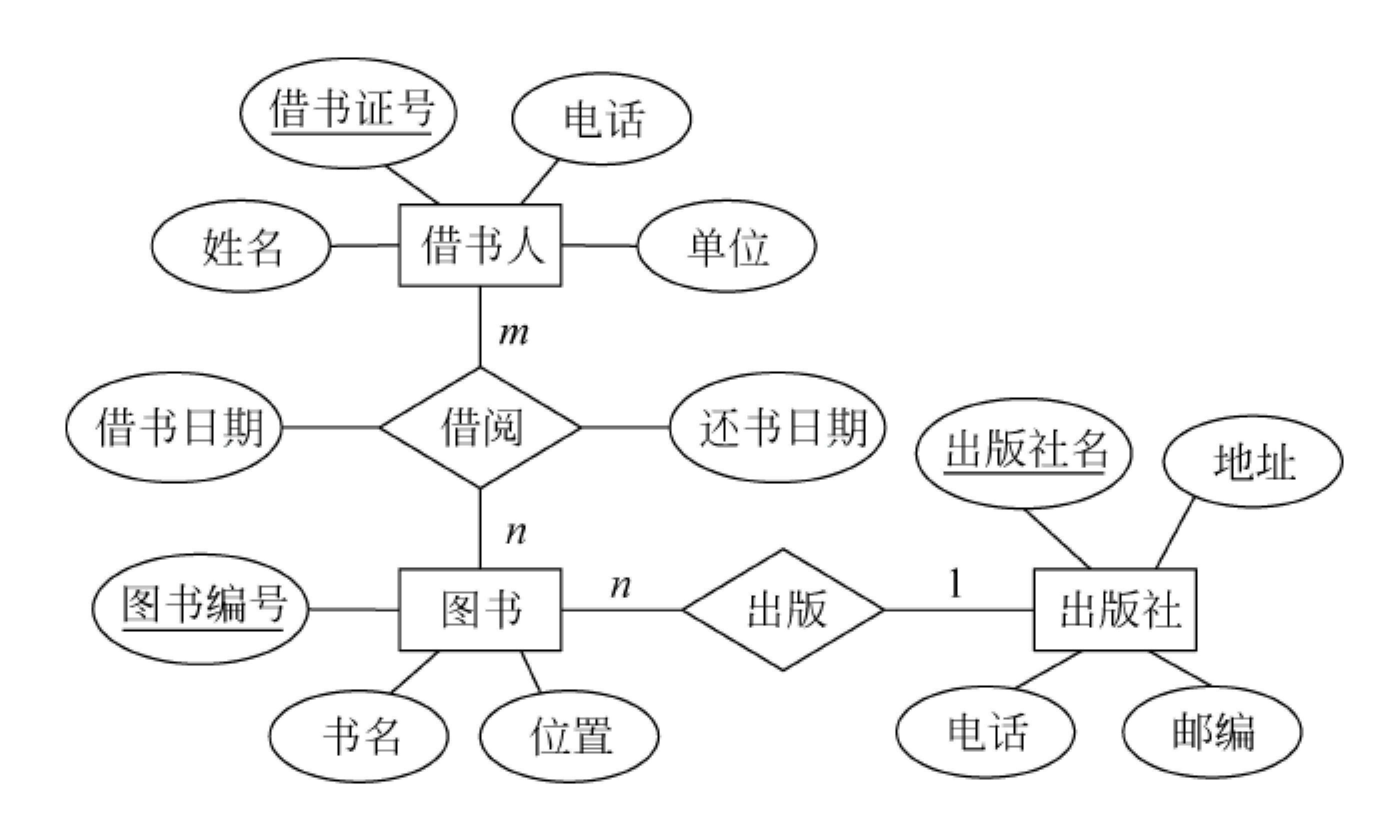


图 4.17　图书借阅信息管理系统 E-R 图

4.6　逻辑结构设计

概念结构设计所得的 E-R 模型是对用户需求的一种抽象的表达形式，它独立于任何一种具体的数据模型，因而也不能为任何一个具体的 DBMS 所支持。为了能够建立起最终的物理系统，还需要将概念结构进一步转化为某一 DBMS 所支持的数据模型，然后根据逻辑设计的准则、数据的语义约束、规范化理论等对数据模型进行适当的调整和优化，形成合理的全局逻辑结构，并设计出用户子模式。这就是数据库逻辑设计所要完成的任务。

4.6.1　逻辑结构设计的步骤

由于各种 DBMS 产品一般都有许多限制，提供不同的环境与工具，因此，逻辑设计分为如下几步：

(1) 将概念模型向一般关系、网状和层次模型转化；

(2) 将得到的一般关系、网状和层次模型向特定的 DBMS 产品所支持的数据模型转化；

(3) 依据应用的需求和具体的 DBMS 的特征进行调整和完善。

数据库逻辑结构的设计过程如图 4.18 所示。

某些早期设计的应用系统中还在使用网状或层次数据模型，而新设计的数据库应用系

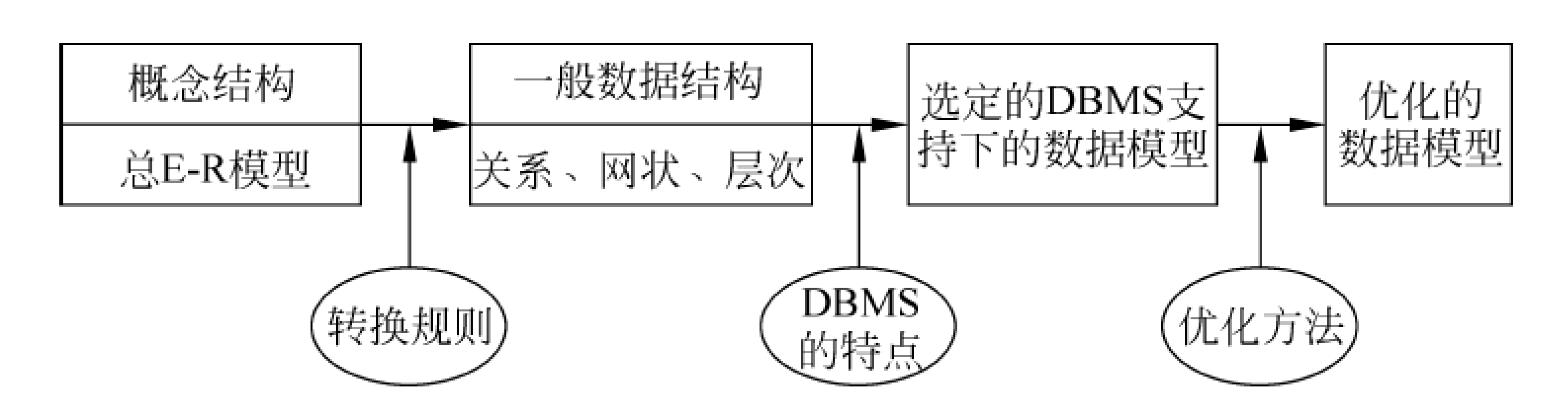


图 4.18 逻辑结构设计的过程

统都普遍采用支持关系数据模型的 RDBMS，所以这里只介绍 E-R 图向关系数据模型的转换原则与方法。

4.6.2 E-R 图向关系模型的转换原则

关系模型的逻辑结构是一组关系模式的集合，而 E-R 图则是由实体、实体的属性和实体之间的联系三个要素组成的。所以将 E-R 图转换为关系模型实际上就是要将实体、实体的属性和实体之间的联系转化为相应的关系模式，下面具体介绍转换的规则。

(1) 一个实体类型转换为一个关系模式。实体的属性就是关系的属性，实体的码就是关系的码。

例题 4.4 将图 4.19 中学生实体和课程实体分别转换成两个关系模式。

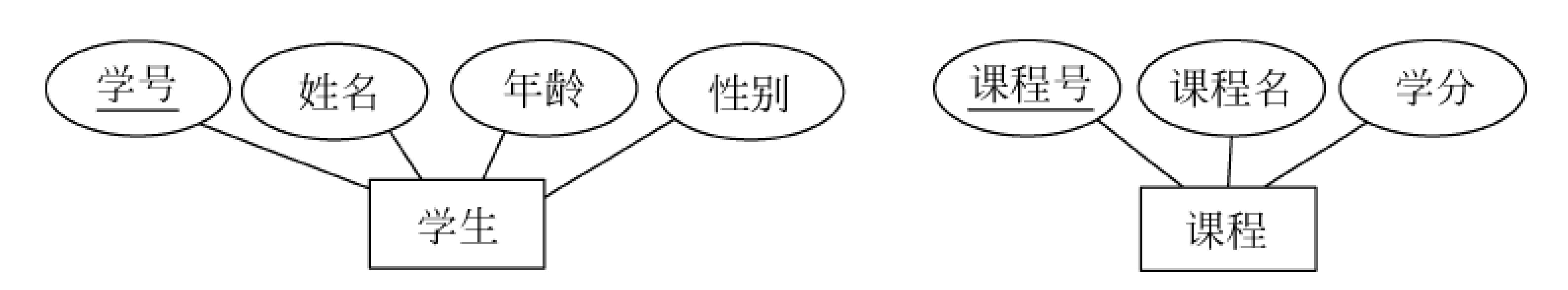


图 4.19 学生和课程实体

学生实体和课程实体分别转换成如下两个关系模式：

学生关系模式(学号,姓名,年龄,性别),学号为关系模式的主码。

课程关系模式(课程号,课程名,学分),课程号为关系模式的主码。

(2) 一个 $m:n$ 联系转换为一个独立的关系模式。与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性。而关系的码为各实体码的组合。

例题 4.5 将图 4.20 中学生选课 E-R 模型转换为相应的关系模式。

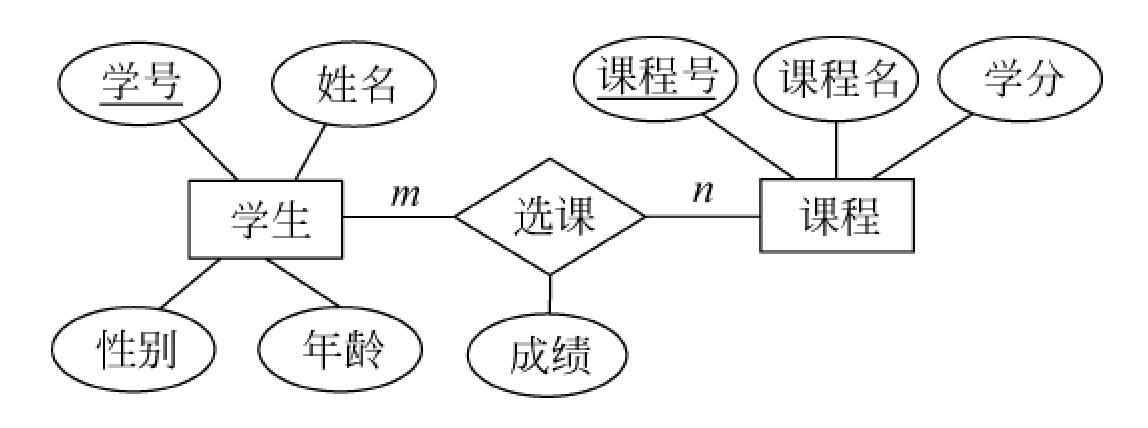


图 4.20 学生选课 E-R 图

将上述 E-R 模型转换为相应的关系模式，先将学生和课程两个实体转换为关系模式，再将这两个实体间的联系转换为关系模式，如下：

学生关系模式(学号，姓名，年龄，性别)，学号为关系模式的主码。

课程关系模式(课程号，课程名，学分)，课程号为关系模式的主码。

选课关系模式(学号，课程号，成绩)，学号和课程号的组合码为关系模式的主码。

(3) 一个 1∶n 联系可以转换为一个独立的关系模式，也可以与 n 端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而该关系的码为 n 端实体的码。如果与 n 端对应的关系模式合并，则只需要将联系本身的属性和 1 端实体的码加入到 n 端对应的关系模式中即可。

例题 4.6　将图 4.21 中班导师指导学生的 E-R 模型转换为相应的关系模式。

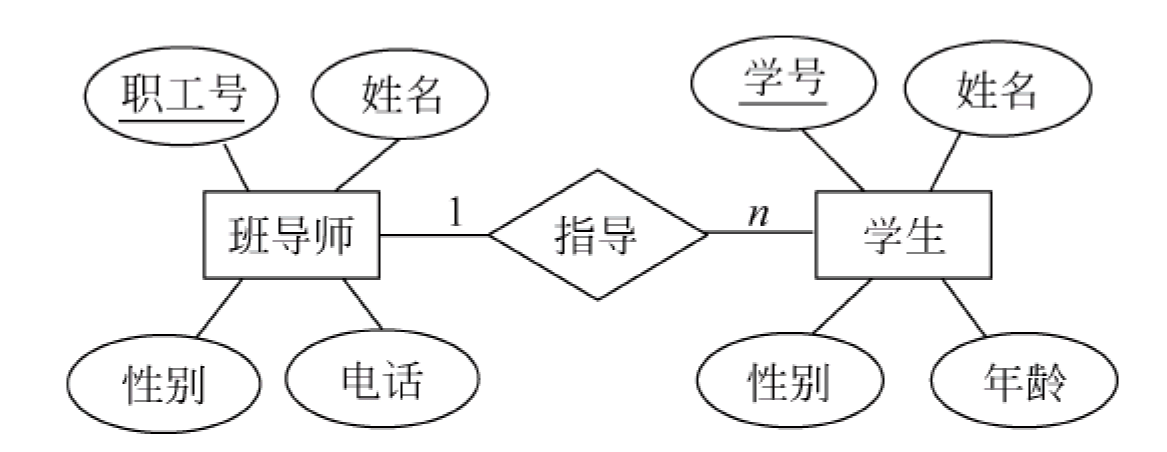


图 4.21　学生指导 E-R 图

将上述 E-R 模型转换为相应的关系模式，先将班导师和学生两个实体转换为关系模式，再将这两个实体间的联系转换为关系模式，如下：

方法一，产生独立的关系模式。

学生关系模式(学号，姓名，年龄，性别)，学号为关系模式的主码。

班导师关系模式(职工号，姓名，性别，电话)，职工号为关系模式的主码。

指导关系模式(学号，职工号)，学号为关系模式的主码。

方法二，与 n 端对应的关系模式合并。

学生关系模式(学号，姓名，年龄，性别，职工号)，学号为关系模式的主码。

班导师关系模式(职工号，姓名，性别，电话)，职工号为关系模式的主码。

(4) 一个 1∶1 联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，每个实体的码均是该关系的候选码。如果与某一端对应的关系模式合并，则需要在该关系模式的属性中加入另一个关系模式的码和联系本身的属性。

例题 4.7　将图 4.22 中的 E-R 模型转换为相应的关系模式。

将上述 E-R 模型转换为相应的关系模式，先将班级和班长两个实体转换为关系模式，再将这两个实体间的联系转换为关系模式，如下：

方法一，产生独立的关系模式。

班级关系模式(班级号，人数)，班级号为关系模式的主码。

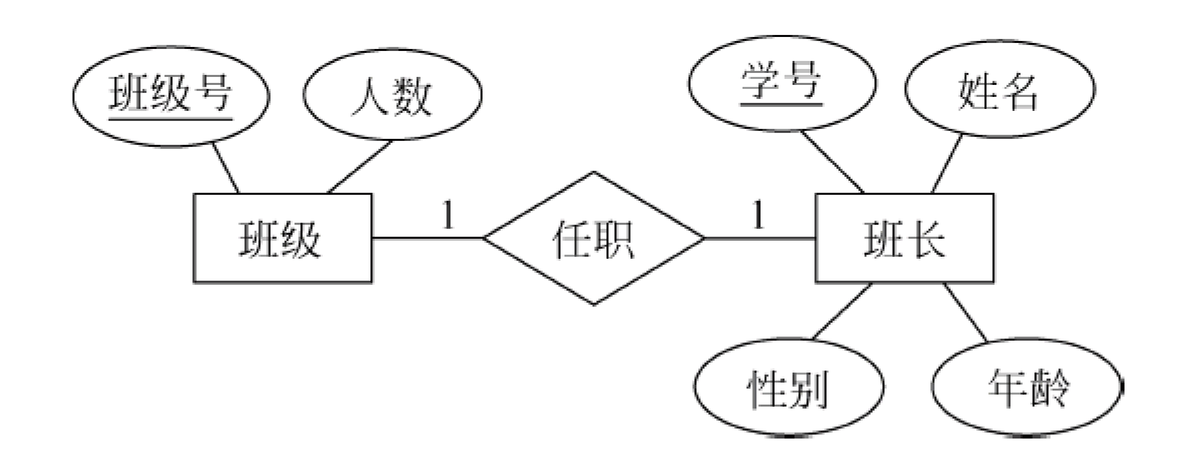


图 4.22　班长任职 E-R 图

班长关系模式(学号,姓名,性别,年龄),学号为关系模式的主码。

任职关系模式(班级号,学号),班级号为关系模式的主码,也可以选学号作为关系模式的主码。

方法二,与任意一端对应的关系模式合并。

班级关系模式(班级号,人数),班级号为关系模式的主码。

班长关系模式(学号,姓名,性别,年龄,班级号),学号为关系模式的主码。

或者

班级关系模式(班级号,人数,学号),班级号为关系模式的主码。

班长关系模式(学号,姓名,性别,年龄),学号为关系模式的主码。

(5) 三个或三个以上实体间的一个多元联系转换为一个关系模式。与该多元联系相连的各实体的码以及联系本身的属性均转换为关系的属性。而关系的码为各实体码的组合。

(6) 同一实体集的实体间的联系,即自联系,也可按上述 1∶1、1∶n 和 m∶n 三种情况分别处理。

(7) 具有相同码的关系模式可合并。

例题 4.8　每个工厂生产多种产品,且每种产品可以在多个工厂中生产,每个工厂按照固定的计划数量生产产品;每个工厂聘用多名职工,且每个职工只能在一个工厂工作,工厂聘用职工有聘用期和工资。工厂的属性有工厂编号、厂名、地址,产品的属性有产品编号、产品名、规格,职工的属性有职工号、姓名。

(1) 根据需求画出 E-R 图,并在 E-R 图中注明实体的属性、联系的类型以及实体标识符。

(2) 将 E-R 图转换成关系模式,并用下划线标出每个关系模式的主码。

例题解析:

(1) 根据题意,建立全局 E-R 图如图 4.23 所示。

(2) 将上述 E-R 图转换成相应的关系模式为:

工厂(工厂编号,厂名,地址)

产品(产品编号,产品名,规格)

职工(职工号,姓名,工厂编号,工资,聘用期)

生产(工厂编号,产品编号,计划数量)

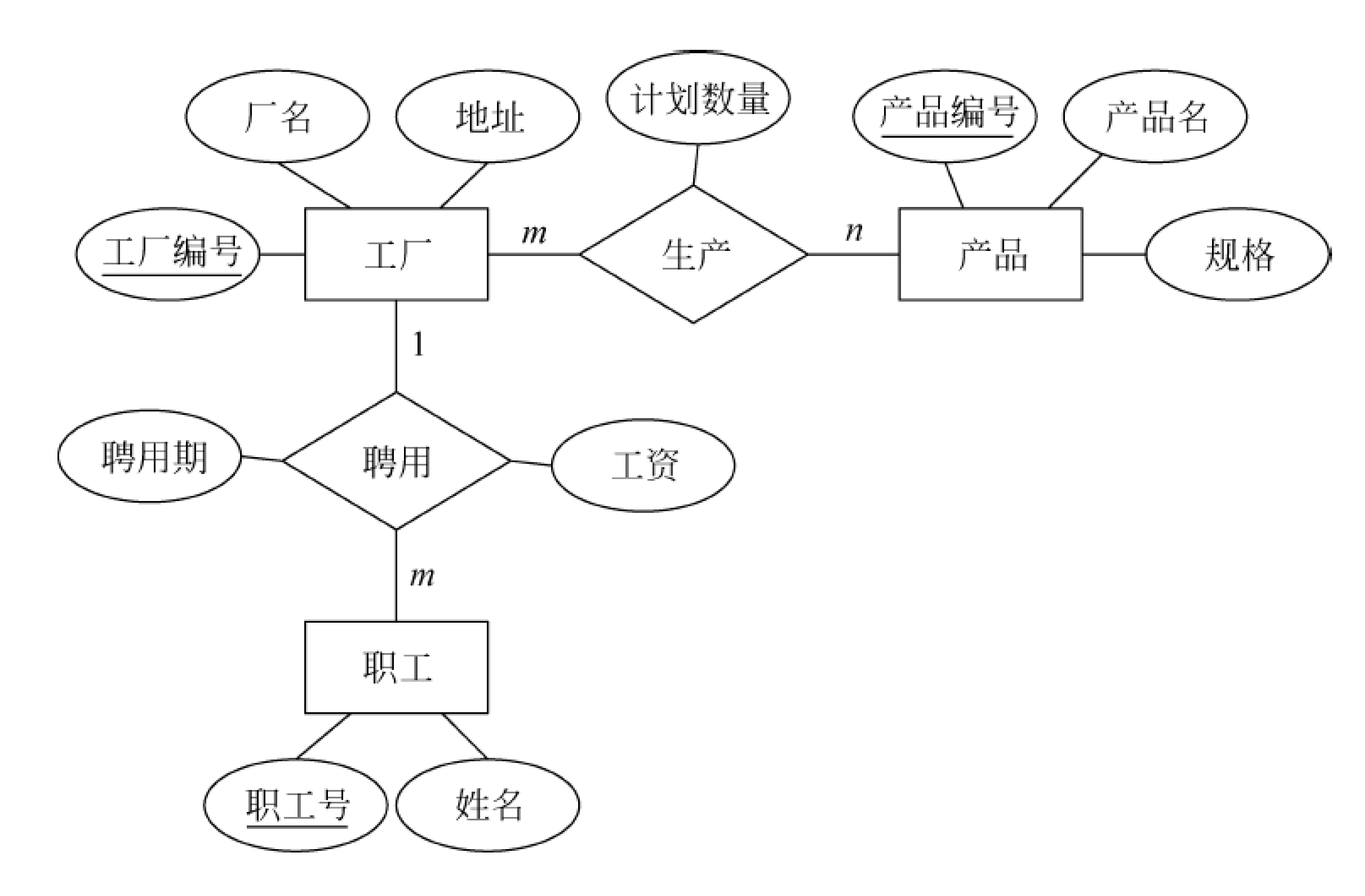


图 4.23　工厂信息管理 E-R 图

例题 4.9　某影院管理中心有如下信息：

影院内有多个放映厅，每个放映厅只属于一个影院；每个放映厅可以放映多部影片，每部影片可以在不同的放映厅放映，每部电影在放映厅放映时有放映时间；每个观众可以观看多部影片，每部影片也可以被多名观众观赏，每个观众观赏影片时都有观看时间。

其中，影院的属性有影院名、地址、电话；放映厅的属性有厅名、规模；电影的属性有许可证号、电影名、类型、时长；观众的属性有身份证号、姓名、年龄。

(1) 根据需求画出 E-R 图，并在 E-R 图中注明实体的属性、联系的类型以及实体的码。

(2) 将 E-R 图转换成关系模式，并用下划线标出每个关系模式的主码。

例题解析：

(1) 根据题意，建立全局 E-R 图如图 4.24 所示。

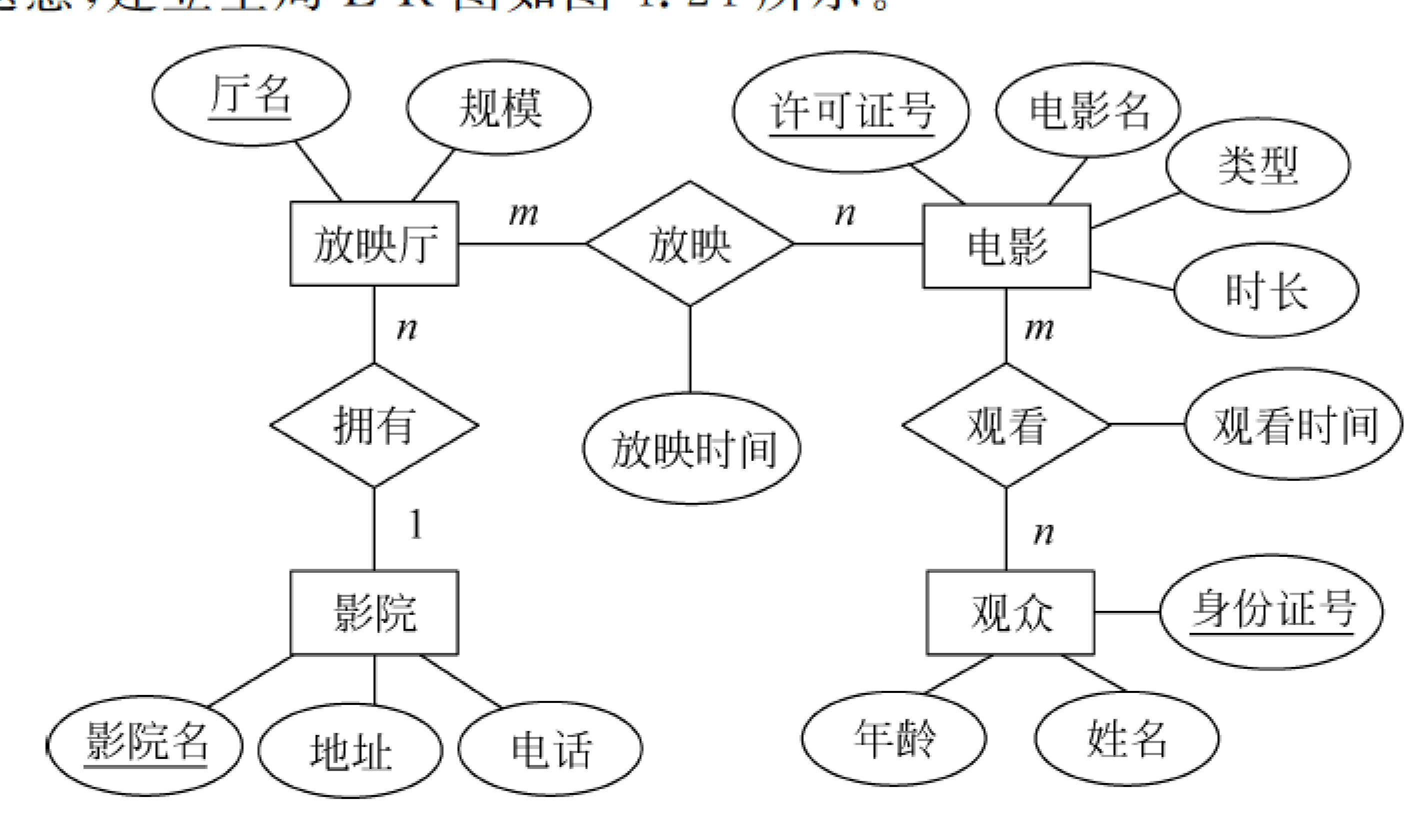


图 4.24　影院管理 E-R 图

(2) 将上述 E-R 图转换成相应的关系模式为：

影院(影院名,地址,电话)

放映厅(厅名,规模,影院名)

电影(许可证号,电影名,类型,时长)

观众(身份证号,姓名,年龄)

放映(厅名,许可证号,放映时间)

观看(身份证号,许可证号,观看时间)

4.6.3 数据模型的优化

关系数据库逻辑设计的结果不是唯一的。在逻辑结构设计的基础上,根据需要对设计结构进行适当的调整和完善,以提高系统的性能。为了进一步提高数据库应用系统的性能,通常以规范化理论为指导,还应该适当地修改、调整数据模型的结构,这就是数据模型的优化。

数据模型的优化方法为：

(1) 确定数据依赖。

(2) 对于各个关系模式之间的数据依赖进行极小化处理,消除冗余的联系。

(3) 按照数据依赖的理论对关系模式逐一进行分析,考查是否存在部分函数依赖、传递函数依赖、多值依赖等,确定各关系模式分别属于第几范式。

(4) 按照需求分析阶段得到的各种应用对数据处理的要求,分析对于这样的应用环境这些模式是否合适,确定是否要对它们进行合并或分解。

(5) 对关系模式进行必要的分解。

如果一个关系模式的属性特别多,就应该考虑是否可以对这个关系进行垂直分解。如果有些属性是经常访问的,而有些属性是很少访问的,则应该把它们分解为两个关系模式。如果一个关系的数据量特别大,就应该考虑是否可以进行水平分解。例如一个论坛中,如果设计时把会员发的主帖和跟帖设计为一个关系,则在帖子量非常大的情况下,就应该考虑把它们分开了。因为显示的主帖是经常查询的,而跟帖则是在打开某个主帖的情况下才查询。又如手机号管理软件,可以考虑按省份或其他方式进行水平分解。

4.7 物理结构设计

数据库物理设计阶段的任务是根据具体计算机系统(DBMS 和硬件等)的特点,为给定的数据库模型确定合理的存储结构和存取方法。所谓“合理”主要有两个含义：一个是要使设计出的物理数据库占用较少的存储空间,另一个对数据库的操作具有尽可能高的速度。

数据库的物理设计通常分为两步：

第一,确定数据库的物理结构,在关系数据库中主要指存取方法和存储结构。

第二，对物理结构进行评价，评价的内容是系统的时间和空间效率。

4.7.1　确定物理结构

1. 确定数据的存储结构

确定数据库存储结构时要综合考虑存取时间、存储空间利用率和维护代价三方面的因素。这三个方面常常是相互矛盾的，例如消除一切冗余数据虽然能够节约存储空间，但往往会导致检索代价的增加，因此必须进行权衡，选择一个折中方案。

2. 设计数据的存取路径

在关系数据库中，选择存取路径主要是指确定如何建立索引。例如，应把哪些域作为次码建立次索引，建立单码索引还是组合索引，建立多少个为合适，是否建立聚集索引等。

3. 确定数据的存放位置

为了提高系统性能，数据应该根据应用情况将易变部分与稳定部分、经常存取部分和存取频率较低部分分开存放。

4. 确定系统配置

DBMS 产品一般都提供了一些存储分配参数，供设计人员和 DBA 对数据库进行物理优化。初始情况下，系统都为这些变量赋予了合理的默认值。但是这些值不一定适合每一种应用环境，在进行物理设计时，需要重新对这些变量赋值以改善系统的性能。

4.7.2　评价物理结构

数据库物理设计过程中需要对时间效率、空间效率、维护代价和各种用户要求进行权衡，其结果可以产生多种方案，数据库设计人员必须对这些方案进行细致的评价，从中选择一个较优的方案作为数据库的物理结构。

评价物理数据库的方法完全依赖于所选用的 DBMS，主要是从定量估算各种方案的存储空间、存取时间和维护代价入手，对估算结果进行权衡、比较，选择出一个较优的合理的物理结构。如果该结构不符合用户需求，则需要修改设计。

4.8　数据库的实施

在进行概念结构设计、逻辑结构设计和物理结构设计之后，设计者对目标系统的结构、功能已经分析得较为清楚了，但这还只是停留在文档阶段。数据库系统设计的根本目的，是为用户提供一个能够实际运行的系统，并保证该系统的稳定和高效。数据库的实施主要包括以下工作：用数据定义语言(DDL)定义数据库结构，组织数据入库，编制与调试应用程序，数据库试运行四个部分。

1. 定义数据库结构

确定了数据库的逻辑结构与物理结构后，就可以用所选用的 DBMS 提供的数据定义语

言(DDL)来严格描述数据库结构。

2. 数据装载

数据库结构建立好后,就可以向数据库中装载数据了。组织数据入库是数据库实施阶段最主要的工作。对于数据量不是很大的小型系统,可以用人工方式完成数据的入库,其步骤为:

1) 筛选数据

需要装入数据库中的数据通常都分散在各个部门的数据文件或原始凭证中,所以首先必须把需要入库的数据筛选出来。

2) 转换数据格式

筛选出来的需要入库的数据,其格式往往不符合数据库要求,还需要进行转换。这种转换有时可能很复杂。

3) 输入数据

将转换好的数据输入计算机中。

4) 校验数据

检查输入的数据是否有误。

对于中大型系统,由于数据量极大,用人工方式组织数据入库将会耗费大量的人力和物力,而且很难保证数据的正确性。因此应该设计一个数据输入子系统由计算机辅助数据的入库工作。

3. 编制与调试应用程序

数据库应用程序的设计应该与数据库设计并行进行。在数据库实施阶段,当数据库结构建立好后,就可以开始编制与调试数据库的应用程序,也就是说,编制与调试应用程序是与组织数据入库同步进行的。调试应用程序时由于数据入库尚未完成,可先使用模拟数据。

4. 数据库试运行

应用程序调试完成,并且已有一小部分数据入库后,就可以开始数据库的试运行。数据库试运行也称为联合调试,其主要工作包括:

1) 功能测试

功能测试即实际运行应用程序,执行对数据库的各种操作,测试应用程序的各种功能。

2) 性能测试

性能测试即测量系统的性能指标,分析是否符合设计目标。

由于在数据库设计阶段,设计者对数据库的评价多是在简化了的环境条件下进行的,因此设计结果未必是最佳的。在试运行阶段,除了对应用程序做进一步的测试之外,重点执行对数据库的各种操作,实际测量系统的各种性能,检测是否达到设计要求。如果在数据库试运行时,所产生的实际结果不理想,则应回过头来修改物理结构,甚至修改逻辑结构。

4.9　数据库的运行和维护

数据库试运行结果符合设计目标后，数据库就可以真正投入运行了。数据库投入运行标志着开发任务的基本完成和维护工作的开始，并不意味着设计过程的终结。由于应用环境在不断变化，数据库运行过程中物理存储也会不断变化，对数据库设计进行评价、调整、修改等维护工作是一个长期的任务，也是设计工作的继续和提高。

在数据库运行阶段，对数据库经常性的维护工作主要是由 DBA 完成的，具体包括如下几个方面。

1. 数据库的转储和恢复

定期对数据库和日志文件进行备份，确保一旦发生故障，能利用数据库备份及日志文件备份，尽快将数据库恢复到某种一致性状态，并尽可能减少对数据库的破坏。

2. 数据库的安全性、完整性控制

DBA 必须对数据库安全性和完整性控制负起责任。根据用户的实际需要授予不同的操作权限。另外，由于应用环境的变化，数据库的完整性约束条件也会变化，也需要 DBA 不断修正，以满足用户要求。

3. 数据库性能的监督、分析和改进

目前许多 DBMS 产品都提供了监测系统性能参数的工具，DBA 可以利用这些工具方便地得到系统运行过程中一系列性能参数的值。DBA 应该仔细分析这些数据，通过调整某些参数来进一步改进数据库性能。

4. 数据库的重组织和重构造

数据库运行一段时间后，由于记录的不断增、删、改，会使数据库的物理存储变坏，从而降低数据库存储空间的利用率和数据的存取效率，使数据库的性能下降。这时 DBA 就要对数据库进行重组织，或部分重组织(只对频繁增、删的表进行重组织)。

重构数据库的程度是有限的。若应用环境的设置变化得太大，已无法通过重构造数据库来满足新的需要，或重构的代价太大，则表明现有数据库应用系统的生命周期已经终结，应该重新设计数据库应用系统，启动新数据库应用系统的生命周期。

4.10　本 章 小 结

本章首先介绍了关系数据库规范化理论，给出了函数依赖和范式的相关定义，通过实例给出如何规范关系模型和保证数据完整性。

其次介绍了数据库设计的基础知识，给出了数据库设计的方法和具体步骤。详细介绍了系统规划阶段、需求分析阶段、概念结构设计阶段、逻辑结构设计阶段、物理结构设计阶段、数据库的实施阶段、数据库的运行和维护阶段的目标、方法和应该注意的事项。

其中最重要的两个环节是概念结构设计阶段和逻辑结构设计阶段。重点介绍了在概念结构设计阶段中，将需求分析的结果转化为 E-R 模型的方法；在逻辑结构设计阶段中，将 E-R 图转换成关系模式的转换内容与转换原则。

4.11 课后习题

一、选择题

1. 设计性能较优的关系模式称为规范化，规范化主要的理论依据是(　　)。

   A. 关系运算理论　　B. 关系规范化理论

   C. 关系代数理论　　D. 数理逻辑

2. 下列函数依赖中，(　　)属于平凡函数依赖。

   A. $(X,Y)\rightarrow Z$　　B. $(X,Y)\rightarrow Y$

   C. $X\rightarrow Z$　　D. $Z\rightarrow Y$

3. 在关系模式中，如果属性 X 和 Y 存在 1∶1 的联系，则说明(　　)。

   A. $X\rightarrow Y$　　B. $Y\rightarrow X$

   C. $X\leftarrow\rightarrow Y$　　D. 以上都不对

4. 关系模式中各级范式之间的关系为(　　)。

   A. 1NF⊂2NF⊂3NF⊂BCNF　　B. 3NF⊂2NF⊂1NF⊂BCNF

   C. BCNF⊂3NF⊂2NF⊂1NF　　D. BCNF⊂1NF⊂2NF⊂3NF

5. 关系模式的候选码可以有(　　)。

   A. 0 个　　B. 1 个　　C. 1 个或多个　　D. 多个

6. 下列不属于需求分析阶段工作的是(　　)。

   A. 分析用户活动　　B. 建立数据流图

   C. 建立数据字典　　D. 建立 E-R 图

7. 下列不属于全局 E-R 模型优化时要达到的目的是(　　)。

   A. 实体类型的个数尽可能少

   B. 实体类型所含属性的个数尽可能少

   C. 实体间联系的冗余最小

   D. 实体完整性和参照完整性

8. 在 E-R 模型中，如果有 5 个不同的实体集，存在 2 个 1∶n 联系和 3 个 m∶n 联系，根据 E-R 模型转换为关系模型的规则，该 E-R 图转换为关系模式的数目至少是(　　)。

   A. 5 个　　B. 7 个　　C. 8 个　　D. 10 个

9. 下面有关 E-R 模型向关系模型转换的叙述中，不正确的是(　　)。

   A. 一个实体类型转换为一个关系模式

   B. 一个 1∶1 联系可以转换为一个独立的关系模式，也可以与联系的任意一端实体

所对应的关系模式合并

C. 一个 1∶m 联系可以转换为一个独立的关系模式，也可以与联系的任意一端实体所对应的关系模式合并

D. 一个 m∶n 联系转换为一个独立的关系模式

10. 对数据库的物理结构设计优劣评价的重点是(　　)。

A. 用户界面的友好性　　B. 动态和静态的性能

C. 时间和空间效率　　D. 成本和效益

二、简答题

1. 简述 2NF 与 3NF 的关系。

2. 什么是 E-R 图？构成 E-R 图的基本要素是什么？

3. 简述 E-R 图向关系模型的转换规则。

三、应用题

1. 在某一商业集团数据库中，有一个关系模式为 R(商店编号，商品编号，库存数量，部门编号，部门经理)。这些数据有下列语义：

(1) 每个商店的每种商品只在一个部门销售。

(2) 每个商店的每个部门只有一个部门经理。

(3) 每个商店的每种商品只有一个库存数量。

请回答下列问题：

(1) 根据上述语义，写出关系模式 R 的基本函数依赖。

(2) 写出关系模式 R 的候选码。

(3) 试问关系模式 R 最高已经达到第几范式？给出理由。

(4) 如果关系模式 R 不满足 3NF，请将 R 规范化到 3NF。

2. 某动物园管理中心有如下信息：

动物园有多个笼舍，每个笼舍只属于这一个动物园；每个笼舍可以安置一种动物，每种动物住在一个笼舍里；每名饲养员喂养多种动物，每种动物可由多名饲养员喂养，饲养员每次喂养动物时有一个喂养时间；每个游客可以观赏多种动物，每种动物可以供多名游客观赏。

其中，动物园的属性有：动物园名、地点、电话；笼舍的属性有笼舍编号、规模、位置；动物的属性有动物编号、动物名称、产地、所属科目；饲养员的属性有饲养员编号、姓名、年龄、职位；游客的属性有：身份证号、姓名、性别。

(1) 根据需求画出 E-R 图，并在 E-R 图中注明实体的属性、联系的类型以及实体标识符。

(2) 将 E-R 图转换成关系模式，并用下划线标出每个关系模式的主码。

第 5 章

Oracle 数据库体系结构

数据库体系结构是从某一角度来分析数据库的组成和工作过程,以及数据库如何管理和组织数据的。因此,在开始对 Oracle 进行操作之前,用户还需要理解 Oracle 数据库的体系结构。Oracle 数据库的体系结构主要包括物理存储结构、逻辑存储结构、内存结构和进程结构四个部分。

5.1 物理存储结构

Oracle 数据库的物理存储结构主要包括 3 大类文件:数据文件(*.DBF)、控制文件(*.CTL)和重做日志文件(*.LOG)。

5.1.1 数据文件

数据文件(Data File)是物理存储 Oracle 数据库数据的文件。包括两部分内容:用户数据和系统数据。数据以一种 Oracle 特有的格式被写入到数据文件,其他程序无法读取数据文件中的数据。用户数据是用来存放用户对象(表或者索引等),而系统数据则主要是指数据字典中的数据。一个 Oracle 数据库一般会包含多个数据文件。

数据文件以.DBF 后缀结尾,可以分为系统数据文件、临时数据文件、回退数据文件和用户数据文件等,存放在 oradata/orcl 文件夹下。

5.1.2 控制文件

控制文件(Control File)是一个较小的二进制文件,用于描述和维护数据库的物理结构。控制文件非常重要,它存放有数据库中数据文件和日志文件的相关信息。这些信息包括:

(1) 数据库的名称。

(2) 数据文件和重做日志文件的名称、位置、联机\脱机状态和大小。

(3) 发生磁盘故障或用户错误时,用于恢复数据库的信息(日志序列号,检查点)。

控制文件以.CTL 后缀结尾,并进行了特殊的加密处理,所以不能直接打开。在数据库

的运行过程中，每当出现数据库检查点或修改数据库的结构之后，Oracle 就会修改控制文件的内容。DBA 可以通过 OEM 工具修改控制文件中的部分内容，但不可以人为随意地修改控制文件中的内容，否则会破坏控制文件。

5.1.3　重做日志文件

重做日志文件(Redo Log File)是记录数据库中所有修改信息的文件，一旦数据库出现问题(如数据库服务器死机或突然断电)，可以通过重做日志文件把数据库恢复到一个正确的状态。

在 oradata/orcl 文件夹下有 3 个重做日志文件，以.LOG 后缀结尾，文件初始化大小为 50MB。这 3 个文件也都经过了特殊的加密处理，所以不能直接打开。重做日志文件以循环方式进行写操作。当第一个日志文件内容达到 50MB 大小后，即写满时，数据库管理系统会自动把日志文件切换到第二个日志文件，当第二个日志文件写满时，切换到第三个日志文件，当第三个日志文件写满后，系统会重新切换到第一个日志文件进行写操作。重做日志文件工作原理，如图 5.1 所示。

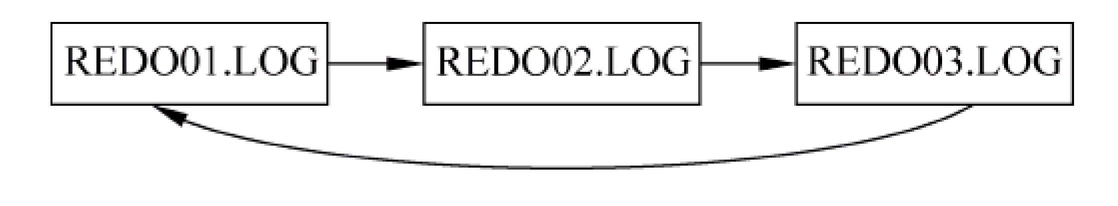


图 5.1　重做日志文件工作原理图

5.1.4　其他文件

在 Oracle 物理存储结构中主要包括上述 3 大类文件，这 3 大类文件缺一不可，少了任何一个，都会造成数据库启动失败。除了 3 大类文件，还包含初始化参数文件、口令文件、归档重做日志文件等物理文件。

(1) 初始化参数文件是在数据库启动和数据库性能调优时使用，记录了数据库各参数的值。该文件在 admin\orcl\pfile 文件夹下，参数文件名为 init.ora。

(2) 口令文件是为了使用操作系统认证 Oracle 用户而设置的。该文件存放在 dbhome_1\database 文件夹下，名称为 PWDorcl.ora。

(3) 归档重做日志文件只有在数据库运行在归档方式时才有，是由 ARCH 归档进程将写满的重做日志文件复制到指定的存储设备时产生的。

例题 5.1　查看 Oracle 11g 数据库安装后的目录结构，确定控制文件、重做日志文件和数据文件的存储位置。

Oracle 11g 数据库安装后的目录结构如图 5.2 所示，物理文件存放在 oradata/orcl 文件夹下。



图 5.2　Oracle 11g 安装后的目录结构

5.2　逻辑存储结构

Oracle 数据库管理系统没有像其他数据库管理系统那样直接操作数据文件，而是引入了一组逻辑存储结构。逻辑存储结构由表空间、段、区和 Oracle 块组成。

引入逻辑存储结构的目的主要有增强 Oracle 数据库的可移植性，降低 Oracle 数据库使用者的操作难度，增加数据的使用安全性。Oracle 数据库逻辑结构和物理结构之间的关系如图 5.3 所示。

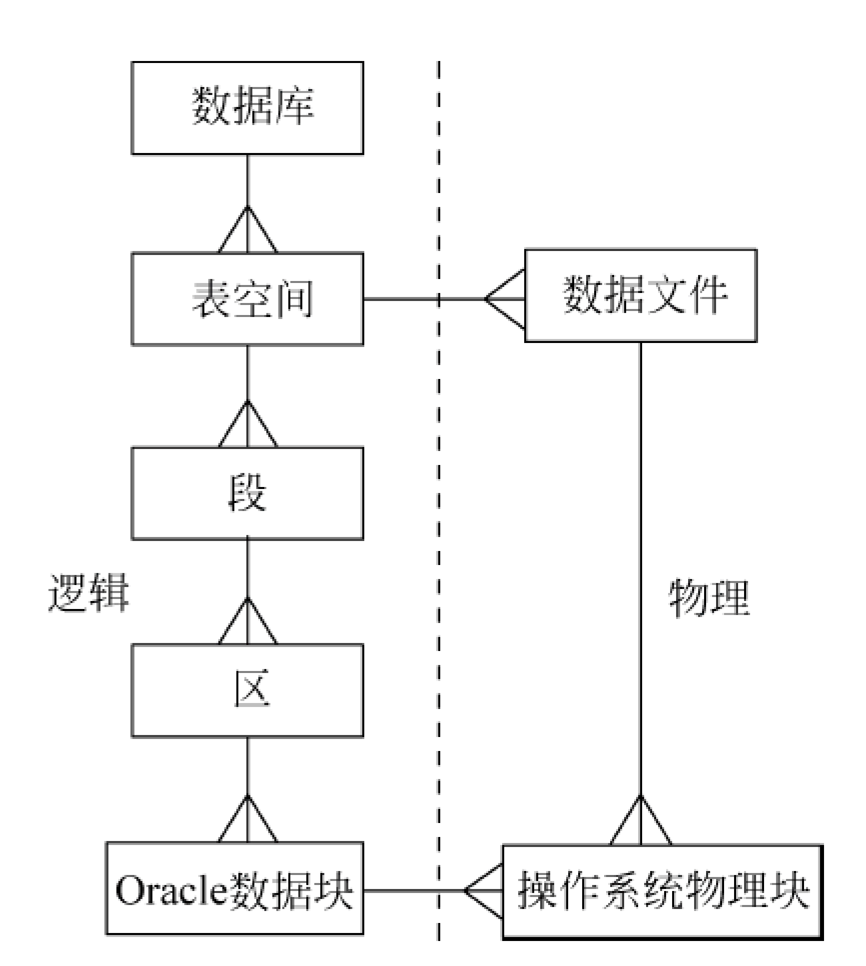


图 5.3　逻辑结构和物理结构之间的关系

一个数据库在逻辑上由多个表空间组成，一个表空间又可以由多个段组成，一个段由若干个区组成，区又是由一组连续的 Oracle 数据块组成。一个表空间可以跨越多个物理结构中的数据文件，一个数据文件只能属于一个表空间所有。一个数据文件由多个操作系统物理块所组成。Oracle 数据块是输入或输出的最小单位，一般由一个或多个操作系统物理块组成。

5.2.1　表空间

表空间(Tablespace)是 Oracle 数据库最大的逻辑结构,一个 Oracle 数据库在逻辑上由多个表空间组成,一个表空间只隶属于一个数据库。表空间在物理上包含一个或多个数据文件。Oracle 数据库中表空间的个数决定了数据文件的最小个数。

表空间大体上可以分为系统表空间和非系统表空间两类。

系统表空间如 SYSTEM 表空间,在系统表空间中主要存放数据字典、过程、函数、触发器等;也可以存储用户的表、索引等。但考虑到 Oracle 系统的效率以及管理上的方便,在系统表空间上不应该存放任何用户的数据。

非系统表空间如为特定项目创建的表空间,它是可以用数据库管理员创建的,在非系统表空间中可以存储一些单独的段,可以是用户的数据段、索引段、还原段和临时段等。引入非系统表空间的主要目的是为了把数据库中的静态数据和动态数据有效地分开,方便磁盘空间的管理。

5.2.2　段

段是比表空间小的下一级逻辑结构。段使用数据文件中的磁盘空间。Oracle 中为了方便数据的管理和维护,提供了多种不同类型的段,主要包括:

(1) 表——也称之为数据段。表段中存储的数据是无序的。Oracle 要求一个表中的所有数据必须放在同一个表空间下。

(2) 索引——可以提高数据的查询速度。当在 Oracle 数据库中创建一个表并指定一个字段为 PRIMARY KEY 或者使用 CREATE INDEX 时,系统自动建立相应的索引段。

(3) 临时段——主要存储临时性的数据。在执行 SQL 或者 PL/SQL 中如果使用了 ORDER BY、GROUP BY 或者 DISTINCT 等关键字时,系统会在内存中进行排序。如果内存排序不下时,需要把中间的结果写到临时段。

(4) 还原段——用来存放事务对数据库的改变。在对数据库中数据或者索引改变时,所有的原始值都存放在还原段中。

(5) 系统引导段——也称为高速缓存段,是数据库创建时由 SQL. BSQ 脚本建立的。该段在打开数据库时自动初始化数据字典高速缓存,无须管理员来维护。

5.2.3　区

区是由一组连续的 Oracle 数据块组成。一个段由若干个区组成,当创建一个段时(如创建一个表时),系统会在相应表空间中找到空闲区(Free Extent)为该段分配空间。当这个段释放或者销毁时,这些释放的区会被添加到所在表空间的空闲区中,以便再次分配和使用。

5.2.4 块

Oracle 数据块是 Oracle 数据库中最小的存储单元。Oracle 数据块是输入或输出的最小单位，一般由一个或多个操作系统物理块组成。其大小由初始化参数文件中的 DB_BLOCK_SIZE 参数设定。

5.3 内存结构

Oracle 中使用的主要内存结构有两个，分别是系统全局区（System Global Area，SGA）和程序全局区（Program Global Area，PGA）。

5.3.1 系统全局区

系统全局区的数据被多个用户共享。当数据库启动时，系统全局区内存被自动分配。它主要包含以下几个内存结构：共享池（Shared Pool）、数据库高速缓冲区（Database Buffer Cache）、重做日志缓冲区（Redo Log Buffer）和其他的一些结构等。

（1）共享池是由库高速缓存（Library Cache）和数据字典高速缓存（Data Dictionary Cache）两部分所组成。系统将 SQL（也可能是 PL/SQL）语句的正文和编译后的代码以及执行计划都放在共享池的库高速缓存中。在进行编译时，系统首先会在共享池中搜索是否有相同的 SQL 或 PL/SQL 语句（正文），如果有就不进行任何后续的编译处理，而是直接使用已存在的编译后的代码和执行计划。当 Oracle 在执行 SQL 语句时，将把数据文件、表、索引、列、用户和其他的数据对象的定义和权限的信息放入数据字典高速缓存。

（2）数据库高速缓冲区主要目的是为了缓存操作的数据，从而减少系统读取磁盘的次数。

（3）重做日志缓冲区主要目的就是为了数据的恢复。Oracle 系统在使用任何 DML 或 DDL 操作时，会先把该操作的信息写入重做日志缓冲区中，之后才对数据高速缓冲区的内容进行修改。

5.3.2 程序全局区

程序全局区是包含单个用户或服务器数据的控制信息的内存区域。是在用户进程连接到 Oracle 数据库并创建一个会话时，由 Oracle 自动分配的。PGA 是非共享区，主要用于在编程时存储变量与数组。会话结束时，释放 PGA。

5.4　进 程 结 构

Oracle 系统中的进程分为以下三类：用户进程、服务器进程和后台进程。

5.4.1　用户进程

当用户在客户端使用应用程序或者 Oracle 工具程序通过网络访问 Oracle 服务器时，客户端会为应用程序分配用户进程。用户进程为该用户单独服务，用户进程不能直接访问数据库。

5.4.2　服务器进程

当用户使用正确的用户名和密码登录成功时，Oracle 系统会在服务器上为该用户创建一个服务器进程。用户进程向服务器进程发请求，服务器进程对数据库进行实际的操作并把所得的结果返回给用户进程。

5.4.3　后台进程

Oracle 数据库的后台进程主要有 5 个，分别是系统监控进程(SMON)、进程监控进程(PMON)、数据库写进程(DBWR)、重做日志写进程(LGWR)和检查点进程(CKPT)。这 5 个进程是必需的，即任何一个没有启动，Oracle 将自动关闭。

(1) 系统监控进程负责在 Oracle 启动时执行数据库恢复，并负责清理不再使用的临时段。

(2) 进程监督进程负责进程的清理工作。主要工作包括回滚用户当前的事务、释放用户所加的所有表一级和行一级的锁、释放用户所有的其他资源等。

(3) 数据库写进程负责将数据库高速缓冲区中的脏缓冲区中的数据写到数据文件上。

(4) 重做日志写进程负责将重做日志缓冲区的记录按顺序地写到重做日志文件中。

(5) 检查点进程用来减少执行 Oracle 恢复所需的时间。引入校验点就是为了提高系统的效率。因为所有到校验点为止的变化了的数据都已写到了数据文件中，这部分数据不用再恢复。

5.5　数据库例程

每一个运行的 Oracle 数据库都对应一个 Oracle 例程(Instance)，也可以称为实例。它是一种访问数据库的机制，由系统全局区和一些后台进程组成。

一个数据库可以由多个实例打开，但任何时刻一个实例只能打开一个数据库。多个实例可以同时运行在同一个机器上，它们彼此访问各自独立的物理数据库。

当实例启动之后，Oracle 会把这个实例以及其对应的物理数据库关联起来，这个过程称为“加载”(Mounting)。这个时候数据库将处于准备打开的状态，数据库在打开之后只有管理员才能够将其关闭，普通用户是无权关闭数据库的。

例题 5.2 练习例程的启动与关闭。

启动数据库例程的命令为 STARTUP，执行 STARTUP 命令时，显示的信息如图 5.4 所示。

```
SQL> STARTUP
ORACLE 例程已经启动。

Total System Global Area  770019328 bytes
Fixed Size                  1374780 bytes
Variable Size             301991364 bytes
Database Buffers          461373440 bytes
Redo Buffers                5279744 bytes
数据库装载完毕。
数据库已经打开。
```

图 5.4 例程的启动

关闭数据库例程的命令为 SHUTDOWN，执行 SHUTDOWN 命令时，显示的信息如图 5.5 所示。

```
SQL> SHUTDOWN
数据库已经关闭。
已经卸载数据库。
ORACLE 例程已经关闭。
```

图 5.5 例程的关闭

5.6 本章小结

本章首先介绍了 Oracle 体系结构的组成，它们分别是物理存储结构、逻辑存储结构、内存结构以及进程结构四个部分。

其次介绍了上述四个组成部分。物理结构中包括控制文件、重做日志文件、数据文件等、逻辑结构中包括表空间、段、区和 Oracle 块、内存结构中包括系统全局区和程序全局区、进程结构中包括用户进程，服务器进程和后台进程。

最后介绍了数据库例程，每一个运行的 Oracle 数据库都对应一个 Oracle 例程。

5.7 课后习题

一、选择题

1. 下列不属于 Oracle 数据库物理结构包括的文件是(　　)。

A. 控制文件　B. 重做日志文件　C. 数据文件　D. 安装文件

2. 下列选项中，哪个不是 Oracle 逻辑结构中的组成部分？（　　）

A. 表空间　B. 区　C. 数据文件　D. Oracle 数据块

3. Oracle 数据库中的数据文件是以什么后缀结尾？（　　）

A. .DBF　B. .CTL　C. .EXE　D. .PDF

4. 下列哪一项是 Oracle 数据库中最小的存储分配单元？（　　）

A. 表空间　B. 段　C. 区　D. Oracle 块

5. 在系统全局区 SGA 中，哪部分内存区域可以用来进行数据恢复操作？（　　）

A. 数据缓冲区　B. 日志缓冲区　C. 共享池　D. 大池

二、简答题

1. 简述 Oracle 数据库逻辑结构和物理结构的关系。

2. 简述 SGA 的组成。

第 6 章

PL/SQL 概述

6.1 PL/SQL 简介

6.1.1 PL/SQL 的定义

PL/SQL 是 Procedure Language & Structured Query Language 的缩写。从名字中能够看出,PL/SQL 包含了两类语句:过程化语句和 SQL 语句。它与 C、Java 等语言一样关注于处理细节,因此可以用来实现比较复杂的业务逻辑。

PL/SQL 通过增加了用在其他过程性语言中的结构来对 SQL 进行了扩展,把 SQL 语言的易用性、灵活性同过程化结构融合在一起。

6.1.2 PL/SQL 的优点

使用 PL/SQL 可以编写具有很多高级功能的程序,虽然通过多个 SQL 语句可能也能实现同样的功能,但是相比而言,PL/SQL 具有更为明显的一些优点:

(1) PL/SQL 可以包含一个或多个程序块,能够使一组 SQL 语句的功能更具模块化程序特点。

(2) PL/SQL 采用了过程性语言控制程序的结构。

(3) PL/SQL 可以对程序中的错误进行自动处理,使程序能够在遇到错误的时候不会被中断。

(4) PL/SQL 能运行在任何 Oracle 环境中(不论它的操作系统和平台),在其他 Oracle 能够运行的操作系统上,无须修改代码,具有较好的可移植性。

(5) PL/SQL 集成在数据库中,调用更快。

(6) PL/SQL 将整个语句块发给服务器,这个过程在单次调用中完成,从而降低了网络拥挤,有助于提高程序性能。

6.1.3 PL/SQL 块结构

PL/SQL 程序的基本结构是块。所有的 PL/SQL 程序都是由块组成的,一般由三部分组成:声明部分、可执行部分和错误处理部分。

PL/SQL 的块结构如下所示。

```
[DECLARE]
/* 声明部分      -- 这部分包括 PL/SQL 变量、常量、游标、用户自定义异常等的定义 */
BEGIN
/* 可执行部分    -- 这部分包括 SQL 语句及过程化的语句,这部分是程序的主体 */
[EXCEPTION]
/* 错误处理部分 -- 这部分包括错误处理语句   */
END;
```

在上面的块结构中,只有可执行部分是必须的,声明部分和错误处理部分都是可选的。块结构中的执行部分至少要有一个可执行语句。

PL/SQL 块可以嵌套使用,对块的嵌套层数没有限制。

嵌套块结构如下所示：

```
[DECLARE]
    ...                         /* 说明部分 */
BEGIN
    ...                         /* 主块的语句执行部分 */
    BEGIN
        ...                     /* 子块的语句执行部分 */
    [EXCEPTION]
        ...                     /* 子块的出错处理程序 */
    END;
[EXCEPTION]
    ...                         /* 主块的出错处理程序 */
END;
```

PL/SQL 支持两种注释样式。

1. 单行注释

如果注释是单行的,或者注释需要嵌入在多行注释中时,可以使用单行注释,单行注释以两个连字符“--”开始,可以扩展到行尾。

例如：

```
v_dname VARCHAR2(20);           -- 这个变量用来处理部门名称
```

2. 多行注释

这些注释以“/*”开始并以“*/”结束,可以跨越多行。建议采用多行注释。

例题 6.1　编写一个 PL/SQL 程序,该程序输出长方形的面积,其中长和宽的值由键盘随机输入。程序运行效果如图 6.1 所示。

例题解析：

```
DECLARE
    v_length NUMBER: = &length;
```

```
    v_width NUMBER: = &width;
    v_area  NUMBER;
BEGIN
    v_area: = v_width * v_length;
    DBMS_OUTPUT.PUT_LINE('该长方形的面积为:'||v_area);
END;
```

```
输入 length 的值:  4
原值    2: v_length NUMBER:=&length;
新值    2: v_length NUMBER:=4;
输入 width 的值:  3
原值    3: v_width NUMBER:=&width;
新值    3: v_width NUMBER:=3;
该长方形的面积为:12

PL/SQL 过程已成功完成。
```

图 6.1 计算长方形面积的程序运行效果

说明：&length 表示从键盘输入一个值，给临时变量 length，而后 length 把接收到的值再传给 v_length。&width 表示从键盘输入一个值，给临时变量 width，而后 width 把接收到的值再传给 v_width。在执行程序之前，需要使用 set serveroutput on 命令设置环境变量 serveroutput 为打开状态，从而使得 PL/SQL 程序能够在 SQL * plus 中输出结果。

6.2 PL/SQL 变量

PL/SQL 中可以使用标识符来声明变量、常量、游标、用户自定义的异常等，并在 SQL 语句或过程化的语句中使用。

6.2.1 标识符定义

PL/SQL 程序设计中的标识符定义与 SQL 的标识符定义的要求相同。要求和限制有：

(1) 不能超过 30 个字符。

(2) 首字符必须为字母。

(3) 字母不区分大小写。

(4) 不能使用 SQL 保留字。

(5) 对标识符的命名最好遵循实际项目中相关命名规范。采用的命名规范要求变量以"v_"开头，常量以"c_"开头，以标识符用途来为其命名。

例如，v_sname 表示一个处理名字的变量，c_birthday 表示一个处理出生日期的常量。

6.2.2 声明语法

PL/SQL 中出现的变量在 DECLARE 部分定义，语法如下：

```
变量名 [CONSTANT] 数据类型 [NOT NULL][ := | DEFAULT PL/SQL 表达式]
```

说明：

(1) 常量的值是在程序运行过程中不能改变的，而变量的值是可以在程序运行过程中不断变化的。声明常量时必须加关键字 CONSTANT，常量在声明时必须初始化，否则在编译时会出错。例如：

```
c_pi CONSTANT NUMBER(8,7) := 3.1415926;
```

如果没有后面的"：= 3.1415926"是没有办法通过编译的。

(2) 如果一个变量没有进行初始化，它将默认被赋值为 NULL。如果使用了非空约束(NOT NULL)，就必须给这个变量赋一个值。在语句块的执行部分或者异常处理部分也要注意不能将 NULL 赋值给被限制为 NOT NULL 的变量。例如：

```
v_flag VARCHAR2(20) NOT NULL := 'true';
```

而不能是

```
v_flag VARCHAR2(20) NOT NULL;
```

声明标识符时，要注意每行声明一个标识符，这样代码可读性更好，也更易于维护。例如：

```
v_sname VARCHAR2(20);
v_dept VARCHAR2(20);
```

而不能是

```
v_sname,v_dept VARCHAR2(20);
```

(3) 初始化变量可以用"：="或"DEFAULT"，如果没有设置初始值，则变量默认被赋值为 NULL；如果变量的值能够确定，最好对变量进行初始化。

(4) 变量名称不要和数据库中表名或字段名相同，否则可能会产生意想不到的结果，另外程序的维护也更加复杂。

建议的命名方法如表 6.1 所示。

表 6.1　变量命名规范

标识符	命名规则	例子
变量	v_name	v_age
常量	c_name	c_pi
游标类型变量	name_cursor	student_cursor
异常类型变量	e_name	e_too_many
记录类型变量	name_record	course_record

6.2.3 数据类型

在 PL/SQL 出现的所有变量和常量都需要指定一个数据类型。下面介绍一些常用的数据类型,有标量类型、参考类型、LOB 类型和用户自定义类型。

1. 标量类型

可以分为以下四种。

1) 数值型

NUMBER [(precision, scale)]:可存储整数或实数值,这里 precision 是精度,即数值中所有数字位的个数,scale 是刻度范围,即小数点右边数字位的个数,精度的默认值是 38。

2) 字符型

CHAR [(maximum_length)] :描述定长的字符串,如果实际值不够定义的长度,系统将以空格填充。在 PL/SQL 中最大长度的是 32 767,长度默认值为 1。

VARCHAR2 (maximum_length):描述变长的字符串。在 PL/SQL 中最大长度是 32 767,没有默认值。

3) 日期型

常用的日期类型为 DATE。日期默认格式为 DD-MON-YY,分别对应日、月、年。例如:

```
v_d1   DATE;
v_d2   DATE : = '13 - 9 月 - 2010';
```

4) 布尔型

存储逻辑值 TRUE 或 FALSE。例如:

```
v_b1 BOOLEAN;
v_b2 BOOLEAN : =  FALSE;
v_b3 BOOLEAN : =  TRUE;
```

2. 参考类型

参考类型分为两种:%TYPE 和%ROWTYPE。

1) %TYPE 类型

定义一个变量,其数据类型可以与已经定义的某个数据变量的类型相同,或者与数据库表的某个列的数据类型相同,这时可以使用%TYPE。例如:

```
v_a1   NUMBER;
v_a2   v_a1 % TYPE;                -- v_a2 参照自 v_a1 变量的类型
v_cname   course.cname % TYPE;              -- v_cname 参照自 COURSE 课程表中 cname 列的数据类型
```

2) %ROWTYPE 类型

定义一个变量的类型参照自基本表或视图中记录的类型、或游标的结构类型,这时可以使用%ROWTYPE。例如:

```
v_sc    sc%ROWTYPE;              --v_sc 参照自 SC 选课信息表中记录的类型
```

说明：由于 v_sc 能代表 SC 选课信息表中的某一条记录类型，所以在访问该记录中某个特定字段时，可以通过“变量名.字段名”的方式调用。例如，向 sc 表中插入一条学生选课信息。

```
DECLARE
    v_sc    sc%ROWTYPE;
BEGIN
    v_sc.sno:='11432001';
    v_sc.cno:='c1';
    v_sc.grade:=90;
    INSERT  INTO  sc  VALUES(v_sc.sno,v_sc.cno,v_sc.grade);
END;
```

3. LOB 类型

用于存储大的数据对象的类型。Oracle 目前主要支持 BFILE、BLOB、CLOB 及 NCLOB 类型。

1）BFILE

存放大的二进制数据对象，这些数据文件不放在数据库里，而是放在操作系统的某个目录里，数据库的表里只存放文件的目录。

2）BLOB

存储大的二进制数据类型。变量存储大的二进制对象的位置。大二进制对象的大小<=4GB。

3）CLOB

存储大的字符数据类型。每个变量存储大字符对象的位置，该位置指到大字符数据块。大字符对象的大小<=4GB。

4）NCLOB

存储大的 NCHAR 字符数据类型。每个变量存储大字符对象的位置，该位置指到大字符数据块。大字符对象的大小<=4GB。

4. 用户自定义类型

根据用户自己的需要，用现有的 PL/SQL 标量类型组合成一个用户自定义的类型。例如，定义用户自定义的数据类型 STUDENT_TYPE。

```
CREATE OR REPLACE TYPE STUDENT_TYPE  AS OBJECT(
    sno    CHAR(8),              --学生学号
    sname VARCHAR2(20),          --学生姓名
    age    INT(3));              --学生年龄
```

引用用户自定义的数据类型。例如：

```
v_stu STUDENT_TYPE;
```

6.2.4 变量赋值

在 PL/SQL 程序中可以通过两种方式给变量赋值。

1. 直接赋值

```
变量名:= 常量或表达式;
```

例如:

```
v_num NUMBER:= 3;
```

2. 通过 SELECT..INTO 赋值

```
SELECT 字段 INTO 变量名
```

例如:

```
SELECT sname,age INTO v_sname, v_age
FROM student
WHERE sno = '11432001';
```

例题 6.2 编写一个 PL/SQL 程序,输出学号为 11432001 的学生姓名和出生年份。程序运行效果如图 6.2 所示。

```
学号为11432001的学生姓名为:陈一,出生年份:1998

PL/SQL 过程已成功完成。
```

图 6.2 查询学生信息的程序运行效果

例题解析:

```
DECLARE
    v_sname student.sname%TYPE;
    v_birth NUMBER;
BEGIN
    SELECT sname,2015 - age INTO v_sname,v_birth FROM student WHERE sno =  '11432001';
    DBMS_OUTPUT.PUT_LINE('学号为 11432001 的学生姓名为:'||v_sname||'出生年份:'||v_birth);
END;
```

6.3 PL/SQL 运算符和函数

6.3.1 PL/SQL 中的运算符

和任何其他的编程语言一样,PL/SQL 有一组运算符,可以分为 3 类:算术运算符、关系运算符和逻辑运算符。

1. 算术运算符

算术运算符执行算术运算，算术运算符有+(加)、-(减)、*(乘)、/(除)、**(指数)和||(连接)。

其中，+(加)和-(减)运算符也可用于对 DATE(日期)数据类型的值进行运算。

2. 关系运算符

关系运算符(又称为比较运算符)用于测试两个表达式值满足的关系，其运算结果为逻辑值 TRUE 和 FALSE。关系运算符有以下几种。

(1) =(等于)、<>或!=(不等于)、<(小于)、>(大于)、>=(大于等于)、<=(小于等于)。

(2) BETWEEN…AND…(检索两值之间的内容)。

(3) IN(检索匹配列表中的值)。

(4) LIKE(检索匹配字符样式的数据)。

(5) IS NULL(检索空数据)。

3. 逻辑运算符

逻辑运算符用于对某个条件进行测试，运算结果为 TRUE 和 FALSE。逻辑运算符有：

(1) AND(两个表达式为真则结果为真)。

(2) OR(只要有一个表达式为真则结果为真)。

(3) NOT(取相反的逻辑值)。

6.3.2 PL/SQL 中的函数

在 PL/SQL 中支持所有 SQL 中的单行数字型的函数、单行字符型的函数、数据类型的转换函数、日期型的函数和其他各种函数，但不支持聚合函数(如 AVG、COUNT、MIN、MAX、SUM 等)。在 PL/SQL 块内只能在 SQL 语句中使用聚合函数。

例题 6.3　举例说明 PL/SQL 块中可以引用的各种函数。

```
v_date: = SYSDATE;
v_name: = UPPER('John');
v_grade: = AVG(grade);          /* 错误,不能在有赋值运算符的语句中使用聚合函数 */
SELECT AVG(grade) INTO v_grade FROM sc;                       /* 正确,这是 SQL 语句 */
```

6.4 PL/SQL 条件结构

PL/SQL 条件结构是根据一条语句或表达式的结果执行另一个操作或一条语句。条件结构分为 IF 条件语句与 CASE 条件语句。

6.4.1 IF 条件语句

语法如下：

```
IF 条件 1 THEN
    语句体 1;
[ELSIF 条件 2 THEN
    语句体 2;]
    …
[ELSE
    语句体 n;]
END IF;
```

说明：如果条件 1 成立，就执行语句体 1 中的内容，否则判断条件 2 是否成立，如果条件 2 成立，执行语句体 2 的内容，以此类推。如果所有条件都不满足，则执行 ELSE 中语句体 n 的内容。

注意：每个 IF 语句以相应的 END IF 语句结束，IF 语句后必须有 THEN 语句，IF…THEN 后不跟语句结束符“;”，一个 IF 语句最多只能有一个 ELSE 语句。条件是一个布尔型的变量或表达式。IF 条件语句最多只能执行一个条件分支，执行之后跳出整个语句块。

例题 6.4 编写一个 PL/SQL 程序，根据某一个学生的平均成绩，判断学生获得的奖学金等级，并输出结果。学号由键盘随机输入，输出等级说明：

(1) 如果平均成绩>85 分，则输出此同学的平均成绩，一等奖学金。

(2) 如果平均成绩在 75～85 分之间，则输出此同学的平均成绩，二等奖学金。

(3) 否则，输出此同学的平均成绩，无奖学金。

程序运行效果如图 6.3 所示。

```
输入 sno 的值:  11432002
原值     2: v_sno student.sno%TYPE:=&sno;
新值     2: v_sno student.sno%TYPE:=11432002;
此同学平均成绩为:75.25,二等奖学金

PL/SQL 过程已成功完成。
```

图 6.3 奖学金等级程序运行效果

例题解析：

```
DECLARE
    v_sno student.sno%TYPE: = &sno;
    v_grade sc.grade%TYPE;
BEGIN
    SELECT AVG(grade) INTO v_grade FROM sc WHERE sno = v_sno;
    IF v_grade > 85 THEN
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为:'||v_grade||',一等奖学金');
ELSIF v_grade >= 75 THEN
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为:'||v_grade||',二等奖学金');
    ELSE
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为:'||v_grade||',无奖学金');
    END IF;
```

```
END;
```

6.4.2　CASE 条件语句

CASE 条件语句又可以有两种写法：含 SELECTOR（选择符）的 CASE 语句和搜索 CASE 语句。

1. 含 SELECTOR（选择符）的 CASE 语句

语法如下：

```
CASE SELECTOR
    WHEN  表达式 1  THEN  语句序列 1;
    [WHEN  表达式 2  THEN  语句序列 2;]
    ...
    [WHEN  表达式 N  THEN  语句序列 N;]
    [ELSE  语句序列 N+1;]
END CASE;
```

说明：SELECTOR 可以是变量或者表达式。当 SELECTOR 和表达式 1 所得到的结果相等时，执行语句序列 1 的内容，以此类推，当 SELECTOR 和所有表达式的结果都不相等时，执行 ELSE 后面的语句 N+1。

例题 6.5　用 CASE 语句判断 v_grade 变量的值是否等于 A、B、C、D、E，并分别处理。程序运行效果如图 6.4 所示。

```
输入 grade 的值:  'B'
原值    2: v_grade VARCHAR2(8):=&grade;
新值    2: v_grade VARCHAR2(8):='B';
良好

PL/SQL 过程已成功完成。
```

图 6.4　成绩等级运行效果

例题解析：

```
DECLARE
    v_grade VARCHAR2(8): = &grade;
BEGIN
    CASE v_grade
        WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('优秀');
        WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('良好');
        WHEN 'C' THEN DBMS_OUTPUT.PUT_LINE('中等');
        WHEN 'D' THEN DBMS_OUTPUT.PUT_LINE('及格');
        WHEN 'E' THEN DBMS_OUTPUT.PUT_LINE('不及格');
        ELSE  DBMS_OUTPUT.PUT_LINE('成绩等级输入非法!');
    END CASE;
```

```
END;
```

2. 搜索 CASE 语句

语法如下：

```
CASE
    WHEN  搜索条件 1  THEN  语句序列 1;
    WHEN  搜索条件 2  THEN  语句序列 2;
    …
    WHEN  搜索条件 N  THEN  语句序列 N;
    [ELSE  语句序列 N+1;]
END CASE;
```

说明：当搜索条件1得到的结果为TRUE时，执行语句序列1的内容，以此类推。当所有搜索条件都不满足时，执行ELSE后面的语句序列N+1。

例题 6.6 将例题6.4中的题目改为CASE语句再实现一次。根据某一个学生的平均成绩，判断学生获得的奖学金等级，并输出结果。学号由键盘随机输入，输出等级说明：

（1）如果平均成绩>85分，则输出此同学的平均成绩，一等奖学金。

（2）如果平均成绩在75～85分之间，则输出此同学的平均成绩，二等奖学金。

（3）否则，输出此同学的平均成绩，无奖学金。

程序运行效果如图6.5所示。

```
输入 sno 的值:  11432003
原值     2: v_sno student.sno%TYPE:=&sno;
新值     2: v_sno student.sno%TYPE:=11432003;
此同学平均成绩为:78,二等奖学金

PL/SQL 过程已成功完成。
```

图 6.5 奖学金等级程序运行效果

例题解析：

```
DECLARE
    v_sno student.sno%TYPE: = &sno;
    v_grade sc.grade%TYPE;
BEGIN
    SELECT AVG(grade) INTO v_grade FROM sc WHERE sno = v_sno;
    CASE
        WHEN v_grade > 85 THEN
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为:'||v_grade||',一等奖学金');
        WHEN v_grade >= 75 THEN
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为:'||v_grade||',二等奖学金');
        ELSE
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为:'||v_grade||',无奖学金');
    END CASE;
```

```
END;
```

6.5　PL/SQL 循环结构

PL/SQL 循环结构是重复地执行一条或多条语句，或者循环一定的次数，或者直到满足某一条件时退出。基本形式是以 LOOP 语句作为循环的开始，以 END LOOP 语句作为循环的结束。

循环语句的基本形式有简单循环、WHILE 循环和数字式 FOR 循环三种。

6.5.1　简单循环

简单循环的特点，循环体至少执行一次，其语法如下：

```
LOOP
    语句体;
    [EXIT;]
END LOOP
```

LOOP 和 END LOOP 之间的语句，如果没有终止条件，将被无限次的执行，显然这种死循环是我们要避免的，在使用 LOOP 语句时必须使用 EXIT 语句，强制循环结束。

退出循环的语法有如下两种形式：

(1) EXIT WHEN 条件；

(2) IF 条件 THEN
　　　EXIT；
　　END IF；

例题 6.7　编写一个 PL/SQL 程序，利用简单循环 LOOP 语句实现输出 1～5 之间的立方数。程序运行效果如图 6.6 所示。

```
1的立方数为1
2的立方数为8
3的立方数为27
4的立方数为64
5的立方数为125

PL/SQL 过程已成功完成。
```

图 6.6　简单循环程序运行效果

例题解析：

方法一：

```
DECLARE
```

```
    i  NUMBER: = 1;
BEGIN
    LOOP
        DBMS_OUTPUT.PUT_LINE(i||'的立方数为'||i * i * i);
        i: = i + 1;
        EXIT WHEN i > 5;
    END LOOP;
END;
```

方法二：

```
    DECLARE
i  NUMBER: = 1;
BEGIN
    LOOP
        DBMS_OUTPUT.PUT_LINE(i||'的立方数为'||i * i * i);
        i: = i + 1;
        IF i > 5 THEN
            EXIT;
        END IF;
    END LOOP;
END;
```

6.5.2 WHILE循环

语法如下：

```
WHILE 条件 LOOP
    语句体;
END LOOP;
```

说明：当条件为TRUE时，执行循环体中的内容，如果结果为FALSE，则结束循环。WHILE循环和以上介绍的简单循环相比，是先进行条件判断，因此循环体有可能一次都不执行。

例题6.8 编写一个PL/SQL程序，利用WHILE循环结构求某个数的阶乘，这个数由键盘输入。程序运行效果如图6.7所示。

例题解析：

```
DECLARE
    v_num   NUMBER: = &num;
    v_sum   NUMBER: = 1;
    i  NUMBER: = 1;
```

```
BEGIN
    WHILE i <= v_num    LOOP
        v_sum: = v_sum *  i;
        i: = i + 1;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(v_num||'的阶乘为:'||v_sum);
END;
```

```
输入 num 的值:  6
原值     2:        v_num  NUMBER:=&num;
新值     2:        v_num  NUMBER:=6;
6的阶乘为:720

PL/SQL 过程已成功完成。
```

图 6.7　WHILE 循环程序运行效果

6.5.3　数字式 FOR 循环

语法如下：

```
FOR counter IN [REVERSE] start_range..end_range LOOP
    语句体;
END LOOP;
```

说明：简单 LOOP 循环和 WHILE 循环的循环次数都是不确定的，FOR 循环的循环次数是固定的，counter 是一个隐式声明的变量，不需要在 DECLARE 部分定义。start_range 和 end_range 指明了循环的次数。

注意：如果使用了 REVERSE 关键字，那么循环变量从最大值向最小值迭代。start_range 和 end_range 之间的省略符只能为“..”。

例题 6.9　编写一个 PL/SQL 程序，利用数字式 FOR 循环实现输出 1～5 之间的立方数。程序运行效果如图 6.8 所示。

```
1的立方数为1
2的立方数为8
3的立方数为27
4的立方数为64
5的立方数为125

PL/SQL 过程已成功完成。
```

图 6.8　数字式 FOR 循环正向输出效果

例题解析：

```
BEGIN
    FOR  i  IN  1..5  LOOP
```

```
        DBMS_OUTPUT.PUT_LINE(i||'的立方数为'||i*i*i);
    END LOOP;
END;
```

例题 6.10 编写一个PL/SQL程序，利用数字式FOR循环反向输出1～5之间的立方数。程序运行效果如图6.9所示。

```
5的立方数为125
4的立方数为64
3的立方数为27
2的立方数为8
1的立方数为1

PL/SQL 过程已成功完成。
```

图6.9 数字式FOR循环反向输出效果

例题解析：

```
BEGIN
    FOR  i REVERSE IN  1..5  LOOP
        DBMS_OUTPUT.PUT_LINE(i||'的立方数为'||i*i*i);
    END LOOP;
END;
```

例题 6.11 编写一个PL/SQL程序，利用数字式FOR循环求某个数的阶乘，这个数由键盘输入。程序运行效果如图6.10所示。

```
输入 num 的值:  5
原值     2: v_num NUMBER:=&num;
新值     2: v_num NUMBER:=5;
5的阶乘为:120

PL/SQL 过程已成功完成。
```

图6.10 数字式FOR循环程序运行效果

例题解析：

```
DECLARE
    v_num NUMBER:=&num;
    v_sum NUMBER:=1;
BEGIN
    FOR  i  IN  2..v_num  LOOP
        v_sum:=v_sum* i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(v_num||'的阶乘为:'||v_sum);
END;
```

6.6 实　　验

6.6.1　实验 1　PL/SQL 基本结构

1. 实验目的

(1) 熟悉 PL/SQL 程序块的基本结构。

(2) 掌握块结构中变量的类型、变量的定义、变量的赋值与变量的输出。

2. 实验内容

样本数据库中，学生表 student、课程表 course 和选课表 sc 的数据信息如图 6.11～图 6.13 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 6.11　学生表 student 中的数据

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图 6.12　课程表 course 中的数据

(1) 练习简单的变量定义与赋值。

已知 a=2，b=3，c=a+b，计算 c 的值并输出。

(2) 编写简单的 PL/SQL 程序块，进行变量定义、变量输入，变量赋值与变量输出的练习。

① 计算王五同学的平均成绩并输出。

② 从键盘随机输入学生学号，查询该学生的选课门数并输出。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。按照要求定义相关变量，程序无语法

```
SNO      CNO           GRADE
-------- -------- ----------
11432001 c1               75
11432001 c2               95
11432002 c1               82
11432002 c2               88
11432002 c4               76
11432002 c5               55
11432003 c1               65
11432003 c2               72
11432003 c3               90
11432003 c4               85
11432004 c1               93
11432004 c2               96
11432004 c5               91
11432005 c1               50
11432005 c2               70
11432005 c5               88
11432005 c8               43
11432007 c1               75
11432007 c2               73
11432007 c3               66
11432007 c6               82
11432007 c7               94
11432008 c1               82
11432008 c2               77
11432008 c3               85
11432008 c4               87
11432008 c5               82
11432008 c6               94
11432008 c7               92
11432008 c8               89
11432009 c1               86
11432009 c2
11432009 c8
11432010 c1               73
11432010 c2
11432010 c8               76
```

图 6.13　选课表 sc 中的数据

错误,书写规范,运行无误,并输出相应程序结果为优秀。如果出现错误,则根据错误点数以及结构合理性灵活给分。

6.6.2　实验 2　PL/SQL 条件语句

1. 实验目的

(1) 掌握 PL/SQL 分支结构中的 IF 语句。

(2) 掌握 PL/SQL 分支结构中的 CASE 语句。

2. 实验内容

样本数据库中，学生表 student、课程表 course 和选课表 sc 的数据信息如图 6.14～图 6.16 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 6.14　学生表 student 中的数据

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图 6.15　课程表 course 中的数据

分别使用 IF 语句和 CASE 语句实现：由键盘随机输入学生学号，查找这位同学的选课门数。

(1) 如果选课门数超过 6 个，则输出“此同学的选课门数为：具体个数，你真是个爱学习的好孩子”。

(2) 如果选课门数少于 3 个，则输出“此同学的选课门数为：具体个数，你真是小懒蛋”。

(3) 否则输出“此同学的选课门数为：具体个数，加油吧”。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。按照要求定义相关变量，分别使用 IF 分支语句、CASE 分支语句实现相应控制结构，并输出程序结果。程序无语法错误，书写规范，运行无误为优秀。如果出现错误，则根据错误点数以及结构合理性灵活给分。

6.6.3　实验 3　PL/SQL 循环语句

1. 实验目的

(1) 掌握 PL/SQL 循环结构中的 WHILE 语句。

(2) 掌握 PL/SQL 循环结构中的 FOR 语句。

```
SNO      CNO           GRADE
-------- -------- ----------
11432001 c1               75
11432001 c2               95
11432002 c1               82
11432002 c2               88
11432002 c4               76
11432002 c5               55
11432003 c1               65
11432003 c2               72
11432003 c3               90
11432003 c4               85
11432004 c1               93
11432004 c2               96
11432004 c5               91
11432005 c1               50
11432005 c2               70
11432005 c5               88
11432005 c8               43
11432007 c1               75
11432007 c2               73
11432007 c3               66
11432007 c6               82
11432007 c7               94
11432008 c1               82
11432008 c2               77
11432008 c3               85
11432008 c4               87
11432008 c5               82
11432008 c6               94
11432008 c7               92
11432008 c8               89
11432009 c1               86
11432009 c2
11432009 c8
11432010 c1               73
11432010 c2
11432010 c8               76
```

图 6.16　选课表 sc 中的数据

2. 实验内容

分别使用 WHILE 语句和 FOR 语句实现：输出 1～20 之间偶数的和。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。按照要求定义相关变量，分别使用 WHILE 循环语句、FOR 循环语句实现相应控制结构，并输出程序结果。程序无语法错误，书写规范，运行无误为优秀。如果出现错误，则根据错误点数以及结构合理性灵活给分。

6.7 本章小结

本章首先介绍了 PL/SQL 的定义和优点、PL/SQL 块的基本结构、PL/SQL 程序中变量的定义、数据类型以及变量的赋值方法。

其次介绍了 PL/SQL 中的 3 类运算符：算术运算符、关系运算符和逻辑运算符，并介绍了 PL/SQL 中所支持的 SQL 函数。

最后介绍了 PL/SQL 程序中的流程控制结构。如条件语句(IF 语句和 CASE 语句),循环语句(简单循环、WHILE 循环和 FOR 循环),并举例说明。

6.8 课后习题

一、选择题

1. 关于 PL/SQL 块结构描述错误的是(　　)。

A. 可执行部分必须有　　B. DECLARE 部分可以没有
C. 异常处理部分可以没有　　D. END 可以没有

2. 下面标识符中正确的是(　　)。

A. 2_test　　B. v_c　　C. sum　　D. AVG

3. 关于循环结构中描述错误的是(　　)。

A. 简单循环需要定义循环变量,通过条件判断何时退出循环
B. FOR 循环中循环变量必须在 DECLARE 中定义,FOR 循环自动判断退出条件
C. WHILE 循环中循环体有可能一次都不执行
D. WHILE 循环先判断条件,只有条件为真时,才执行循环体中的内容

4. PL/SQL 中的循环不包括(　　)。

A. DO…WHILE 循环　　B. WHILE 循环
C. FOR 循环　　D. 简单循环

5. 执行以下语句:

```
DECLARE
    i NUMBER: = 1;
    n NUMBER: = 0;
BEGIN
    FOR  i  IN 2..3  LOOP
        n: = n + i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(n);
END;
```

执行完成后输出的结果是(　　)。

A. 0　　B. 3　　C. 5　　D. 6

二、应用题

1. 从键盘上输入两个数分别作为 FOR 循环的起始条件,并反向输出循环变量的值。

2. 随机输入一个学生学号,判断该同学的平均成绩属于哪一个等级,并输出等级。规则如下:

(1) 当平均成绩>=90 为优秀;

（2）当 90＞平均成绩＞＝80 为良好；
（3）当 80＞平均成绩＞＝70 为中等；
（4）当 70＞平均成绩＞＝60 为及格；
（5）否则为不及格。
（分别用 IF 和 CASE 语句实现，& 表示可以在运行时接受输入值）

第 7 章

异常处理

7.1 异常概述

7.1.1 Oracle 异常处理机制

异常(EXCEPTION)是一种 PL/SQL 标识符,如果运行 PL/SQL 块时出现错误或警告,则会触发异常。当触发异常时,默认情况下会终止 PL/SQL 块的执行。通过在 PL/SQL 块中引入异常处理部分,可以捕捉各种异常,并根据异常出现的情况进行相应的处理。

(1) 编译时错误:指代码不满足特定语法的要求,由编译器发出错误报告。由于编译时错误主要是语法方面的错误,如果不修改程序就无法执行,因此该错误可以由程序员修改。

(2) 运行时错误:指程序运行过程中出现的各种问题,由引擎发出报告。运行时错误是随着运行环境的变化而随时出现的,难以预防,因此需要程序中尽可能地考虑各种可能出现的错误。

Oracle 对运行时错误的处理采用了异常处理机制。

7.1.2 异常的类型

Oracle 运行时的错误可以分为 Oracle 错误和用户自定义错误,与之对应,异常分为预定义异常、非预定义异常和用户自定义异常 3 种。其中预定义异常对应于常见的 Oracle 错误,非预定义异常对应于其他的 Oracle 错误,而用户自定义异常对应于用户自定义错误。

(1) 预定义异常:当 Oracle 错误产生时,与错误对应的预定义异常被自动抛出,通过捕获该异常可以对错误进行处理。

(2) 非预定义异常:即其他标准的 Oracle 错误。用于处理预定义异常所不能处理的 Oracle 错误。

(3) 用户自定义异常:处理与 Oracle 错误无关的其他情况。对这种异常情况的处理,需要用户在程序中定义,然后显式地在程序中将其引发。

7.1.3 异常处理的基本语法

异常处理部分一般放在 PL/SQL 程序块的后半部，具体语法为：

```
EXCEPTION
    WHEN 错误 1 [OR 错误 2] THEN
        语句序列 1;
    WHEN 错误 3 [OR 错误 4] THEN
        语句序列 2;
    ...
WHEN OTHERS THEN
    语句序列 n;
```

说明：在 PL/SQL 块中，当错误发生时，程序控制无条件地转移到当前 PL/SQL 块的异常处理部分。一旦控制转移到异常处理部分，就不能再转到相同块的可执行部分。WHEN OTHERS 从句放置在所有其他异常处理从句的后面，最多只能有一个 WHEN OTHERS 从句。

7.2 预定义异常

每当 PL/SQL 违背了 Oracle 原则或超越了系统依赖的原则就会隐式地产生内部错误。对这种异常情况的处理，无须在程序中定义，由 Oracle 自动将其引发，编程中只需要在 EXCEPTION 异常处理部分按照错误名称处理它们就可以了。经常出现的系统预定义的错误如表 7.1 所示。

表 7.1 系统预定义错误

错误名称	错误号	错误说明
DUP_VAL_ON_INDEX	ORA-00001	唯一值约束被破坏
TIMEOUT_ON_RESOURCE	ORA-00051	在等待资源时发生超时现象
INVALID_CURSOR	ORA-01001	非法的游标操作
NOT_LOGGED_ON	ORA-01012	没有连接到 Oracle
LOGIN_DENIED	ORA-01017	无效的用户名/口令
NO_DATA_FOUND	ORA-01403	没有找到数据
TOO_MANY_ROWS	ORA-01422	SELECT...INTO 语句匹配多行数据
ZERO_DIVIDE	ORA-01476	被零除
INVALID_NUMBER	ORA-01722	转换为一个数字失败
STORAGE_ERROR	ORA-06500	运行时内存不够引发内部的 PL/SQL 错误
PROGRAM_ERROR	ORA-06501	内部 PL/SQL 错误
VALUE_ERROR	ORA-06502	截尾、算术或转换错误

续表

错误名称	错误号	错误说明
ROWTYPE_MISMATCH	ORA-06504	游标变量和 PL/SQL 结果集之间数据类型不匹配
CURSOR_ALREADY_OPEN	ORA-06511	试图打开已存在的游标
ACCESS_INTO_NULL	ORA-06530	试图为 NULL 对象的属性赋值

例题 7.1　验证 ZERO_DIVIDE 异常。编写一个 PL/SQL 程序，计算 x 除以 y 的结果并输出。x 的初始值为 6，y 的初始值为 0。程序运行效果如图 7.1 所示。

```
DECLARE
*
第 1 行出现错误:
ORA-01476: 除数为 0
ORA-06512: 在 line 6
```

图 7.1　程序运行时的异常

例题解析：

```
DECLARE
    x NUMBER:=6;
    y NUMBER:=0;
    z NUMBER;
BEGIN
    z:=x/y;
    DBMS_OUTPUT.PUT_LINE(z);
END;
```

例题 7.2　验证 ZERO_DIVIDE 异常。再编写一个 PL/SQL 程序实现上述功能，若出现异常，则对其进行捕获，并且输出两行提示信息："捕获到了 ZERO_DIVIDE 异常"和"错误，除数不能为 0!"。程序运行效果如图 7.2 所示。

```
捕获到了ZERO_DIVIDE异常
错误, 除数不能为0!

PL/SQL 过程已成功完成。
```

图 7.2　对 ZERO_DIVIDE 异常的捕获

例题解析：

```
DECLARE
    x NUMBER:=6;
    y NUMBER:=0;
    z NUMBER;
BEGIN
    z:=x/y;
```

```
        DBMS_OUTPUT.PUT_LINE(z);
    EXCEPTION
        WHEN ZERO_DIVIDE THEN
          DBMS_OUTPUT.PUT_LINE('捕获到了 ZERO_DIVIDE 异常');
          DBMS_OUTPUT.PUT_LINE('错误,除数不能为 0! ');
    END;
```

7.3 非预定义异常

非预定义异常用于处理与预定义异常无关的 Oracle 错误。使用预定义异常,可以处理的错误是有限的。而当使用 PL/SQL 开发应用程序时,可能还会遇到其他一些 Oracle 错误。例如,在 PL/SQL 程序中,把员工表中员工张三所隶属的部门编号修改为 70。

```
BEGIN
    UPDATE emp SET deptno = 70 WHERE ename = '张三';
END;
/
```

运行结果如图 7.3 所示。

```
第 1 行出现错误:
ORA-02291: 违反完整约束条件 (SYSTEM.SYS_C0011118) - 未找到父项关键字
ORA-06512: 在 line 2
```

图 7.3 程序运行时的异常

由于部门表中没有 70 号部门,所以出现了上述错误提示信息。为了提高 PL/SQL 程序的健壮性,应该在 PL/SQL 应用程序中合理地处理这些 Oracle 错误,此时就需要使用非预定义异常。

非预定义异常的处理步骤分为定义异常、关联错误和异常处理三个步骤。

1. 定义异常

在 DECLARE 部分定义异常,异常的类型为 EXCEPTION。

定义异常的语法:

```
异常名 EXCEPTION;
```

例如:

```
DECLARE
    my_exception EXCEPTION;
```

2. 关联错误

在 DECLARE 部分,将其定义好的异常情况与标准的 Oracle 错误联系起来,使用 PRAGMA EXCEPTION_INIT 语句。

关联错误的语法：

```
PRAGMA EXCEPTION_INIT(异常名,错误代码);
```

例如：

```
DECLARE
    my_exception EXCEPTION;
    PRAGMA EXCEPTION_INIT(my_exception, -02291);
```

3. 异常处理

在 EXCEPTION 部分处理，和预定义异常的处理方式一致。如果没有 EXCEPTION 部分，则由系统处理异常。

处理异常的语法：

```
WHEN 异常名 THEN 处理语句;
```

例如：

```
EXCEPTION
    WHEN my_exception THEN
        DBMS_OUTPUT.PUT_LINE('违反完整性约束,未找到父项关键字');
```

例题 7.3　在 PL/SQL 程序中，删除学生表中王五同学的基本信息。如果不能删除则关联并处理异常，输出两行提示信息："捕获到预定义异常 fk_exception"和"选课表中存在该同学的选课记录，该同学的基本信息无法删除！"。程序运行效果如图 7.4 所示。

```
捕获到非预定义异常fk_exception
选课表中存在该同学的选课记录，该同学的基本信息无法删除!

PL/SQL 过程已成功完成。
```

图 7.4　非预定义异常处理

例题解析：

```
DECLARE
    fk_exception EXCEPTION;
    PRAGMA EXCEPTION_INIT(fk_exception, -02292);
BEGIN
    DELETE FROM student WHERE sname = '王五';
EXCEPTION
    WHEN fk_exception THEN
        DBMS_OUTPUT.PUT_LINE('捕获到非预定义异常 fk_exception');
        DBMS_OUTPUT.PUT_LINE('选课表中存在该同学的选课记录,该同学的基本信息无法删除!');
END;
```

7.4 用户自定义异常

预定义异常和非预定义异常都是由 Oracle 判断的异常错误。在实际的程序开发中，为了实施具体的业务逻辑规则，程序开发人员往往会根据这些逻辑规则自定义一些异常。当用户进行操作时违反了这些规则，就会引发一个自定义异常，从而中断程序的正常执行，并转到自定义异常处理部分。

用户自定义异常的处理步骤分为定义异常、触发异常和异常处理三个步骤。

1. 定义异常

在 DECLARE 部分定义异常，异常的类型为 EXCEPTION。

定义异常的语法：

```
异常名 EXCEPTION;
```

例如：

```
DECLARE
    my_exception EXCEPTION;
```

2. 触发异常

在 BGEIN 部分，当一个设定条件满足时，可以显式通过 RAISE 语句来触发自定义异常。

触发异常的语法：

```
RAISE 异常名;
```

例如：

```
BEGIN
    IF v_salary = 0 THEN
        RAISE my_exception;
    END IF;
```

3. 异常处理

在 EXCEPTION 部分处理，和系统预定义异常的处理方式一致。如果没有 EXCEPTION 部分，则由系统处理异常。

处理异常的语法：

```
WHEN 异常名 THEN 处理语句;
```

例如：

```
EXCEPTION
```

```
    WHEN my_exception THEN
        DBMS_OUTPUT.PUT_LINE('员工的工资为 0');
```

例题 7.4　编写一个 PL/SQL 程序，输出姚二同学选修 c5 课程的成绩。如果成绩小于 60 分，则触发用户自定义异常，输出提示信息："成绩不及格，请准备补考！"。

例题解析：

```
DECLARE
    v_grade sc.grade% TYPE;
    e EXCEPTION;
BEGIN
    SELECT grade INTO v_grade FROM sc,student WHERE student.sno = sc.sno AND sname = '姚二' AND
cno = 'c5';
    DBMS_OUTPUT.PUT_LINE('姚二同学选修 c5 课程的成绩是'||v_grade);
    IF v_grade < 60 THEN
        RAISE e;
    END IF;
EXCEPTION
    WHEN e THEN
        DBMS_OUTPUT.PUT_LINE('成绩不及格,请准备补考!');
END;
```

例题 7.5　编写一个 PL/SQL 程序，从键盘上随机输入某个员工的姓名，输出该员工的编号和工资。

（1）如果雇员不存在，触发系统预定义异常 NO_DATA_FOUND，输出："查无此人！"。

（2）如果雇员存在，但工资小于 2000 元，触发用户自定义异常，输出："工资太低，需要涨工资！"。

（3）如果触发了其他异常，输出："未知的错误！"。

（4）如果雇员存在，且工资＞＝2000 元，输出该雇员的编号和工资。

例题解析：

```
DECLARE
    v_ename emp.ename%type: = &p_ename;
    v_empno emp.empno%TYPE;
    v_salary emp.salary%TYPE;
    e EXCEPTION;
BEGIN
    SELECT empno,salary INTO v_empno, v_salary FROM emp WHERE ename = v_ename;
    IF v_salary < 2000 THEN
    RAISE e;
    ELSE
        DBMS_OUTPUT.PUT_LINE('编号为:'||v_empno||'工资为:'||v_salary);
    END IF;
EXCEPTION
```

```
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('查无此人!');
        WHEN e THEN
            DBMS_OUTPUT.PUT_LINE('工资太低,需要涨工资!');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('未知的错误!');
END;
```

7.5 实 验

7.5.1 实验 1 系统预定义异常

1. 实验目的

(1) 熟悉异常处理的语法和原则。

(2) 掌握系统预定义异常的处理步骤。

2. 实验内容

样本数据库中,学生表 student、课程表 course 和选课表 sc 的数据信息如图 7.5～图 7.7 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 7.5 学生表 student 中的数据

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图 7.6 课程表 course 中的数据

SNO	CNO	GRADE
11432001	c1	75
11432001	c2	95
11432002	c1	82
11432002	c2	88
11432002	c4	76
11432002	c5	55
11432003	c1	65
11432003	c2	72
11432003	c3	90
11432003	c4	85
11432004	c1	93
11432004	c2	96
11432004	c5	91
11432005	c1	50
11432005	c2	70
11432005	c5	88
11432005	c8	43
11432007	c1	75
11432007	c2	73
11432007	c3	66
11432007	c6	82
11432007	c7	94
11432008	c1	82
11432008	c2	77
11432008	c3	85
11432008	c4	87
11432008	c5	82
11432008	c6	94
11432008	c7	92
11432008	c8	89
11432009	c1	86
11432009	c2	
11432009	c8	
11432010	c1	73
11432010	c2	
11432010	c8	76

图 7.7　选课表 sc 中的数据

验证系统预定义异常的作用：

（1）编写带有异常处理的 PL/SQL 程序，从键盘上随机输入某门课程名称，输出该课程的学分。若该课程不存在，则触发系统预定义异常 NO_DATA_FOUND，输出“查无此课！”。

（2）编写一个 PL/SQL 程序，输出学生表中杨珺婷同学所在的系别名称，若找不到该同学的系别名称，则引发系统预定义异常，输出两行提示信息：“捕获到了 NO_DATA_FOUND 异常”和“SELECT 语句未找到相应的记录！”。若查询到多个系别名称，则引发另一个系统预定义异常，输出两行提示信息：“捕获到了 TOO_MANY_ROWS 异常”和“SELECT 语句检索到多行数据！”。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。程序无语法错误，书写规范，正确定义 EXCEPTION 处理部分，进行异常的判定触发，运行无误为优秀。如果出现错误，则根据错误点数以及结构合理性灵活给分。

7.5.2 实验2 用户自定义异常

1. 实验目的

(1) 熟悉异常处理的语法和原则。

(2) 掌握用户自定义异常的处理步骤。

2. 实验内容

样本数据库中,学生表student、课程表course和选课表sc的数据信息如图7.8～图7.10所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图7.8 学生表student中的数据

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图7.9 课程表course中的数据

根据用户自定义异常的步骤,编写异常处理程序:

(1) 查找"侦查系"的学生人数,依据人数进行系别大小的判定:

① 如果人数>200,则输出:it is big。

② 如果人数>100,则输出:it is middle。

③ 否则,则输出:it is small。

若此系人数少于10,则触发用户自定义异常,并输出:此系应扩大招生。

(2) 从键盘上随机输入某个学生的学号,查询该学生的不及格课程数。

① 不及格课程数>3时,则触发异常;当此异常发生时,输出"留级"。

② 当不及格课程数为2或3门时,则触发异常;当此异常发生时,输出"跟班试读"。

③ 其余情况,正常输出不及格课程数。

```
SNO      CNO           GRADE
-------- -------- ----------
11432001 c1               75
11432001 c2               95
11432002 c1               82
11432002 c2               88
11432002 c4               76
11432002 c5               55
11432003 c1               65
11432003 c2               72
11432003 c3               90
11432003 c4               85
11432004 c1               93
11432004 c2               96
11432004 c5               91
11432005 c1               50
11432005 c2               70
11432005 c5               88
11432005 c8               43
11432007 c1               75
11432007 c2               73
11432007 c3               66
11432007 c6               82
11432007 c7               94
11432008 c1               82
11432008 c2               77
11432008 c3               85
11432008 c4               87
11432008 c5               82
11432008 c6               94
11432008 c7               92
11432008 c8               89
11432009 c1               86
11432009 c2
11432009 c8
11432010 c1               73
11432010 c2
11432010 c8               76
```

图 7.10　选课表 sc 中的数据

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。程序无语法错误，书写规范，正确定义 EXCEPTION 处理部分，进行异常的判定触发，运行无误为优秀。如果出现错误，则根据错误点数以及结构合理性灵活给分。

7.6　本章小结

本章首先介绍了 Oracle 中的异常处理机制、异常的基本概念以及按照不同的错误对异常进行分类。

其次介绍了系统预定义异常、非预定义异常和用户自定义异常的处理步骤。Oracle 为用户提供了大量的在 PL/SQL 中使用的预定义异常，以检查用户代码失败的一般原因。对这种异常情况的处理，无须在程序中定义，由 Oracle 自动将其引发。对非预定义异常的处理，需要用户在程序中定义，然后由 Oracle 自动将其引发。对用户自定义异常的处理，需要

用户在程序中定义,然后由用户引发异常。

7.7 课后习题

一、选择题

1. PL/SQL 语句块中,当 SELECT... INTO 语句不返回任何数据行时,将抛出异常(　　)。

A. NO_DATA_FOUND　　B. VALUE_ERROR

C. DUP_VAL_INDEX　　D. TOO_MANY_ROWS

2. PL/SQL 语句块中,当 SELECT... INTO 语句返回多行记录时,将抛出异常(　　)。

A. NO_DATA_FOUND　　B. VALUE_ERROR

C. DUP_VAL_INDEX　　D. TOO_MANY_ROWS

3. 关于出错处理,下列叙述错误的是(　　)。

A. 可以有多个 WHEN OTHERS 从句

B. 可以在块中定义多个出错处理,每一个出错处理包含一组语句

C. 在块中必须以关键字 EXCEPTION 开始一个出错处理

D. 将 WHEN OTHERS 从句放置在所有其他出错处理从句的后面

4. 关于用户自定义异常的步骤中,不包括(　　)。

A. 定义异常　　B. 触发异常　　C. 分析异常　　D. 处理异常

5. 用户自定义异常必须使用(　　)语句引发。

A. IF　　B. WHEN　　C. EXCEPTION　　D. RAISE

二、简答题

简述自定义异常的处理步骤。

三、应用题

1. 编写带有异常处理的 PL/SQL 程序:输出选修课程号为 c1 的学生人数。

(1) 若人数少于 10 人,则该课程是不允许开设的,此时触发一个异常,输出提示:“选修人数太少,无法开课”。

(2) 若人数超过 50 人,选修人数超过了最大的选修人数上限,也是不允许的,也触发一个异常,则输出提示:“选修人数过多,需要增加授课教师”。

2. 编写带有异常处理的 PL/SQL 程序:修改员工表 EMP 中某位雇员的工资(从键盘随机输入员工编号),如果工资小于 1500 元,涨 50%;工资在 1500~2000 元之间,涨 30%;在 2001~3000 元之间,涨 10%;超过 3000 元,则触发一个异常,输出“工资太高,不需涨薪”。

第 8 章 游　　标

8.1　游标的定义

在通过 SELECT 语句查询时，返回的结果通常是多行记录组成的集合。这对于程序设计语言而言，并不能够处理以集合形式返回的数据，为此，SQL 提供了游标机制。游标充当指针的作用，使应用程序设计语言一次只能处理查询结果中的一行。

游标在字面上的理解就是游动的光标。用数据库语言来描述：游标是映射在结果集中一行数据上的位置实体，有了游标，用户就可以使用游标来访问结果集中的任意一行数据，提取当前行的数据后，即可对该行数据进行操作，当使用 SQL 语句时，Oracle 服务器打开一个内存区域，用来分析和执行语句，游标就是这块区域的句柄或者指针，借助游标，PL/SQL 能够控制这块区域中记录的访问。

在 Oracle 中，游标分为显式游标和隐式游标。

8.2　显 式 游 标

显式游标是由程序员定义和命名的，并且在块的执行部分中通过特定语句操纵的内存工作区。当 SELECT 返回多条记录时，必须显式地定义游标以处理每一行。

8.2.1　显式游标的处理步骤

显式游标的处理分为定义游标、打开游标、取值到变量和关闭游标四个步骤。

(1) 定义游标：在 DECLARE 说明部分定义游标。

定义游标时需要定义游标的名字，并将该游标和一个 SELECT 语句相关联。这时候相当于给游标所能操作的内存区域做个规划。数据并没有加载到内存区域。

定义游标的语法：

```
CURSOR  游标名[(参数名 1  数据类型[,参数名 2  数据类型...])]
IS  SELECT  语句;
```

说明：CURSOR 是游标的关键字，游标名满足标识符的要求。数据类型可以是任意的 PL/SQL 可以识别的类型，如标量类型、参考类型等。当数据类型是标量类型时，不能定义类型的长度。游标中的 SELECT 语句不用接 INTO 语句。

（2）打开游标：在语句执行部分或者出错处理部分打开游标。

打开游标就是在程序运行时，游标接受实际参数值后，执行游标所对应的 SELECT 语句，将其查询结果放入内存工作区，并且指针指向工作区的首部。

打开游标的语法：

```
OPEN  游标名  [(实际参数值 1[,实际参数值 2...])];
```

（3）将当前行结果提取到 PL/SQL 变量中：在语句执行部分或者出错处理部分提取结果。

取值工作是将游标工作区中的当前指针所指行的数据取出，放入到指定的变量中。系统每执行一次 FETCH 语句只能取一行，每次取出数据之后，指针顺序下移一行，使下一行成为当前行。

由于游标工作区中的记录可能有多行，所以通常使用循环执行 FETCH 语句，直到整个查询结果集都被返回。

取值到变量的语法：

```
FETCH  游标名 INTO  变量 1 [,变量 2...];
```

或

```
FETCH  游标名 INTO  PL/SQL_RECORD;   /*记录类型变量*/
```

（4）关闭游标：在语句执行部分或者出错处理部分关闭游标。

显式打开的游标需要显式关闭。游标关闭后，系统释放与该游标关联的资源，并使该游标的工作区变成无效。关闭以后不能再对游标进行 FETCH 操作，否则会触发一个 INVALID_CURSOR 错误。如果需要可以重新打开。

关闭游标的语法：

```
CLOSE  游标名;
```

8.2.2 显式游标的属性

游标由于每次都是以相同的方式处理内存工作区中的一条记录，为了能对所有记录处理，需要和循环结构搭配使用。而循环的开始及退出，必须以游标的属性为依据。显式游标的属性如表 8.1 所示。

表 8.1　显式游标的属性

游标属性	描　　述
游标名%ISOPEN	值为布尔型,如果游标已打开,取值为 TRUE
游标名%NOTFOUND	值为布尔型,如果最近一次 FETCH 操作没有返回结果,则取值为 TRUE
游标名%FOUND	值为布尔型,如果最近一次 FETCH 操作没有返回结果,则取值为 FALSE;否则取值为 TRUE
游标名%ROWCOUNT	值为数字型,值是到当前为止返回的记录数

例题 8.1　通过游标利用简单循环从学生表中取出某一系别的学生姓名和年龄并输出(注:系别名称从键盘随机输入)。程序运行效果如图 8.1 所示。

```
输入 p_dept 的值:  '侦查系'
原值    2: v_dept student.dept%type:=&p_dept;
新值    2: v_dept student.dept%type:='侦查系';
学生姓名为:陈一,年龄为:17
学生姓名为:姚二,年龄为:20
学生姓名为:张三,年龄为:19

PL/SQL 过程已成功完成。
```

图 8.1　通过游标利用简单循环查询学生信息

例题解析:

```
DECLARE
    v_dept student.dept%type: = &p_dept;
    v_sname student.sname%type;
    v_age student.age%TYPE;
    CURSOR student_cursor IS SELECT sname,age FROM student WHERE dept = v_dept;
BEGIN
    OPEN student_cursor;
    LOOP
    FETCH student_cursor INTO v_sname,v_age;
    EXIT WHEN student_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('学生姓名为:'||v_sname||','||'年龄为:'||v_age);
    END LOOP;
    CLOSE student_cursor;
END;
```

例题 8.2　通过游标利用 WHILE 循环从学生表中取出某一系别的学生姓名和年龄并输出,最后显式该系别的学生人数(注:系别名称从键盘随机输入)。程序运行效果如图 8.2 所示。

例题解析:

```
DECLARE
    v_dept student.dept%type: = &p_dept;
```

```
    CURSOR student_cursor IS SELECT sname,age FROM student WHERE dept = v_dept;
    student_record student_cursor%rowtype;
BEGIN
    OPEN student_cursor;
    FETCH student_cursor INTO student_record;
    WHILE student_cursor%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE('学生姓名为:'||student_record.sname||','||'年龄为:'||student_
record.age);
    FETCH student_cursor INTO student_record;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('学生人数为:'||student_cursor%ROWCOUNT );
    CLOSE student_cursor;
END;
```

```
输入 p_dept 的值:  '公安信息系'
原值    2: v_dept student.dept%type:=&p_dept;
新值    2: v_dept student.dept%type:='公安信息系';
学生姓名为:陈七,年龄为:23
学生姓名为:刘八,年龄为:21
学生人数为:2

PL/SQL 过程已成功完成。
```

图 8.2　通过游标利用 WHILE 循环查询学生信息

8.2.3　游标的 FOR 循环

通常情况下,游标处理数据的步骤可以分为 8 步:

(1) 定义一个游标。

(2) 打开一个游标。

(3) 启动循环。

(4) FETCH 游标到变量。

(5) 检查是否所有的行都返回。

(6) 处理返回的行。

(7) 结束循环。

(8) 关闭游标。

游标的 FOR 循环是一种快捷处理游标的方式,它使用 FOR 循环依次读取内存工作区中的一行数据,当 FOR 循环开始时,游标自动打开(不需要使用 OPEN 方法),每循环一次系统自动读取游标当前行的数据(不需要使用 FETCH),当退出 FOR 循环时,游标被自动关闭(不需要使用 CLOSE)。

使用游标 FOR 循环的时候不需要也不能使用 OPEN 语句,FETCH 语句和 CLOSE 语句,否则会产生错误。

使用游标的 FOR 循环，系统隐式地定义了一个游标名%ROWTYPE 类型的记录变量。把游标所指向当前记录的数据放入到该记录变量中去。

游标 FOR 循环的语法：

```
FOR  记录变量名 IN  游标名 LOOP
语句 1;
语句 2;
...
END LOOP;
```

例题 8.3　利用游标的 FOR 循环从学生表中取出某一系别的学生姓名和年龄，并输出（注：系别名称从键盘随机输入）。程序运行效果如图 8.3 所示。

```
输入 p_dept 的值:  '刑事技术系'
原值    2: v_dept student.dept%type:=&p_dept;
新值    2: v_dept student.dept%type:='刑事技术系';
学生姓名为:李四,年龄为:22
学生姓名为:王五,年龄为:22
学生姓名为:赵六,年龄为:19

PL/SQL 过程已成功完成。
```

图 8.3　利用游标的 FOR 循环查询学生信息

例题解析：

```
DECLARE
    v_dept student.dept%type: = &p_dept;
    CURSOR student_cursor IS SELECT sname,age FROM student WHERE dept = v_dept;
BEGIN
    FOR student_record IN student_cursor LOOP
    DBMS_OUTPUT.PUT_LINE('学生姓名为:'||student_record.sname||','||'年龄为:'||student_
record.age);
    END LOOP;
END;
```

说明：系统会隐式地将 student_record 变量定义为 student_cursor% ROWTYPE 类型。

8.2.4　利用游标操纵数据库

通过使用显式游标，不仅可以一行一行地处理 SELECT 语句的结果，而且可以更新或删除当前游标行的数据。

1. 游标的定义

语法如下：

```
CURSOR 游标名 IS
```

```
SELECT 列 1,列 2…
FROM 表
WHERE 条件
FOR UPDATE [OF column] [NOWAIT]
```

说明：要想通过游标操纵数据库中的数据，在定义游标的查询语句时，必须加上 FOR UPDATE 从句，表示要先对表加锁。此时在游标工作区中的所有行拥有一个行级排他锁，其他会话只能查询，不能更新或删除。OF 子句用来指定要锁定的列。

如果游标查询涉及多张表时，FOR UPDATE 默认情况下会在所有表的记录上拥有行级排他锁。

使用 FOR UPDATE 会给被作用行加锁，如果其他用户已经在被作用行上加锁的话，默认情况下当前用户要一直等待。使用 NOWAIT 选项，可以避免等待锁。一旦其他用户已经在被作用行加锁的话，当前用户会显式系统预定义错误，并退出 PL/SQL 块。

2. 游标的使用

带 WHERE CURRENT OF 从句的 UPDATE、DELETE 语句的语法：

```
DELETE FROM 表 WHERE  CURRENT OF 游标名;
UPDATE 表 SET 列 1 = 值 1,列 2 = 值 2 … WHERE CURRENT OF 游标名;
```

说明：在 UPDATE 或 DELETE 语句中，加上 WHERE CURRENT OF 子句，指定了从游标工作区中取出的当前行需要被更新或删除。

例题 8.4 使用游标更新数据，查询学生表中侦查系学生的基本情况，并输出当前学生的学号、姓名和年龄；如果学生的年龄小于 18 岁，则将数据库中该学生的年龄改成 18 岁。

例题解析：

```
DECLARE
CURSOR student_cursor IS SELECT * FROM student WHERE dept = '侦查系' FOR UPDATE OF age;
BEGIN
    FOR student_record IN student_cursor LOOP
        DBMS_OUTPUT.PUT_LINE(student_record.sno||','||student_record.sname||','||student_
record.age);
        IF student_record.age < 18 THEN
            UPDATE student SET age = 18 WHERE CURRENT OF student_cursor;
        END IF;
    END LOOP;
END;
```

8.2.5 带参数的游标

使用带参数的游标可以提高程序的灵活性。定义显式游标时，加入参数的定义，在使用游标时，对参数输入不同的数值，则游标工作区中所包含的数据也有所不同。

带参数的游标除了定义游标与打开游标时的语法与一般显式游标不同外，其他步骤的

语法都一样。

1. 定义带参数的游标

语法如下：

```
CURSOR 游标名(参数名 1 数据类型[{:=|DEFAULT} 值 ][,参数名 2 数据类型[{:=|DEFAULT}
值 ]...]) IS SELECT 语句;
```

说明：参数的命名满足标识符的命名规则。数据类型可以是标量类型，参考类型等。当是标量类型时，不能指定参数的长度。参数的值一般在 SELECT 语句的 WHERE 子句中使用。

2. 打开带参数的游标

语法如下：

```
OPEN  游标名(&参数 1,&参数 2...);
```

例题 8.5 用带参数游标的简单循环实现从员工表中查询部门号为 20 的员工姓名和工资，并输出。(注：20 为实参)

例题解析：

```
DECLARE
CURSOR emp_cursor (v_deptno NUMBER) IS SELECT ename,salary FROM emp  WHERE deptno = v_deptno;
emp_record emp_cursor%ROWTYPE;
BEGIN
    OPEN emp_cursor(20);
    LOOP
        FETCH  emp_cursor INTO emp_record;
        EXIT  WHEN emp_cursor%NOTfound ;
        DBMS_OUTPUT.PUT_LINE('员工姓名为:'||emp_record.ename||','||'工资为:'||emp_record.
salary);
    END LOOP;
    CLOSE emp_cursor;
END;
```

例题 8.6 用带参数游标的 FOR 循环依此输出每一个部门名称，在部门名称的下面输出该部门的员工姓名和工资，按工资的升序排列。程序运行效果如图 8.4 所示。

例题解析：

```
DECLARE
CURSOR dept_cursor IS SELECT * FROM dept;
CURSOR emp_cursor (v_deptno NUMBER) IS SELECT * FROM emp WHERE  deptno = v_deptno ORDER BY
salary ASC;
BEGIN
FOR dept_record IN dept_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('部门名称为:'||dept_record.dname);
```

```
        FOR emp_record IN emp_cursor(dept_record.deptno) LOOP
DBMS_OUTPUT.PUT_LINE('员工姓名为:'|| emp_record.ename||','||'工资为: '||emp_record.salary);
END LOOP;
END LOOP;
END;
```

```
部门名称为:财务部
员工姓名为:李四,工资为: 3500
员工姓名为:张三,工资为: 4500
员工姓名为:姚二,工资为: 6000
部门名称为:人力资源部
员工姓名为:陈七,工资为: 1800
员工姓名为:赵六,工资为: 2500
员工姓名为:王五,工资为: 5500
部门名称为:销售部
员工姓名为:孙十,工资为: 2400
员工姓名为:张九,工资为: 3500
员工姓名为:刘八,工资为: 6500
部门名称为:研发部
员工姓名为:刘四,工资为: 1800
员工姓名为:崔三,工资为: 3000
员工姓名为:吴二,工资为: 3500
员工姓名为:沈一,工资为: 6500
部门名称为:客服部
员工姓名为:韩六,工资为: 2200
员工姓名为:张五,工资为: 4000

PL/SQL 过程已成功完成。
```

图 8.4　用带参数游标的 FOR 循环实现输出部门及员工信息

8.3　隐式游标

当用户执行 SELECT 语句返回一行记录时,或者执行 DML 语句,如 UPDATE、DELETE、INSERT 操作,则由系统自动地为这些操作设置游标并创建其工作区,这些由系统隐式创建的游标称为隐式游标,隐式游标的名字为 SQL。

对于隐式游标的操作,如定义、打开、取值及关闭操作,都由系统自动地完成,无须用户进行处理。用户只能通过隐式游标的相关属性,来完成相应的操作。

8.3.1　隐式游标的属性

隐式游标的属性和显式游标的属性基本一致,但含义上有所不同,如表 8.2 所示。

表 8.2　隐式游标的属性

属性	属性值	DELETE	UPDATE	INSERT	SELECT
SQL%FOUND	TRUE	成功	成功	成功	有结果
SQL%FOUND	FALSE	失败	失败	失败	没结果
SQL%NOTFOUND	TRUE	失败	失败	失败	没结果

续表

属性	属性值	DELETE	UPDATE	INSERT	SELECT
SQL%NOTFOUND	FALSE	成功	成功	成功	有结果
SQL%ROWCOUNT	行数	删除的行数	修改的行数	插入的行数	1
SQL%ISOPEN	FALSE	FALSE	FALSE	FALSE	FALSE

8.3.2　显式游标与隐式游标的区别

显式游标与隐式游标的区别如表 8.3 所示。

表 8.3　显式游标与隐式游标的比较

显式游标	隐式游标
在程序中显式地定义、打开、关闭。游标有一个名字	当执行插入、更新、删除、以及查询只有一条记录时，由 PL/SQL 内部管理，自动打开和关闭的游标。游标名为 SQL
游标属性的前缀是游标名	游标属性的前缀是 SQL
%ISOPEN 属性有一个有效值，依赖游标的状态	游标属性%ISOPEN 总是 FALSE，因为当语句执行完后立即关闭隐式游标
可以处理任何行。在程序中设置循环过程，每一行都应该显式地取(除非在一个游标的 FOR 循环中)	SELECT...INTO 语句只能处理一行

例题 8.7　随机输入一个员工编号，删除该员工的基本信息，如果操作成功，提示“已删除该员工，删除成功”，否则提示“无法删除该员工，删除失败”。程序运行效果如图 8.5 所示。

```
输入 p_empno 的值:  2008
原值    2:    DELETE FROM emp WHERE empno=&p_empno;
新值    2:    DELETE FROM emp WHERE empno=2008;
无法删除该员工, 删除失败!

PL/SQL 过程已成功完成。

SQL> /
输入 p_empno 的值:  2003
原值    2:    DELETE FROM emp WHERE empno=&p_empno;
新值    2:    DELETE FROM emp WHERE empno=2003;
已删除该员工, 删除成功!

PL/SQL 过程已成功完成。
```

图 8.5　删除某一员工信息

例题解析：

```
BEGIN
    DELETE FROM emp WHERE empno = &p_empno;
    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('无法删除该员工,删除失败! ');
    ELSE
        DBMS_OUTPUT.PUT_LINE('已删除该员工,删除成功! ');
    END IF;
END;
```

8.4 实　　验

8.4.1 实验1 不带参数的游标

1. 实验目的

(1) 熟悉游标的概念和属性。

(2) 掌握游标的处理步骤和不带参数游标的应用。

2. 实验内容

样本数据库中，学生表student、课程表cours、选课表sc、员工表emp和部门表dept的数据信息如图8.6～图8.10所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图8.6　学生表student中的数据

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图8.7　课程表course中的数据

SNO	CNO	GRADE
11432001	c1	75
11432001	c2	95
11432002	c1	82
11432002	c2	88
11432002	c4	76
11432002	c5	55
11432003	c1	65
11432003	c2	72
11432003	c3	90
11432003	c4	85
11432004	c1	93
11432004	c2	96
11432004	c5	91
11432005	c1	50
11432005	c2	70
11432005	c5	88
11432005	c8	43
11432007	c1	75
11432007	c2	73
11432007	c3	66
11432007	c6	82
11432007	c7	94
11432008	c1	82
11432008	c2	77
11432008	c3	85
11432008	c4	87
11432008	c5	82
11432008	c6	94
11432008	c7	92
11432008	c8	89
11432009	c1	86
11432009	c2	
11432009	c8	
11432010	c1	73
11432010	c2	
11432010	c8	76

图 8.8　选课表 sc 中的数据

EMPNO	ENAME	SEX	AGE	JOB	MGR	SALARY	DEPTNO
1001	陈一	男	52	总经理		9500	
2001	姚二	男	47	部门经理	1001	6000	10
2002	张三	男	35	会计	2001	4500	10
2003	李四	女	27	出纳	2001	3500	10
3001	王五	女	38	部门经理	1001	5500	20
3002	赵六	女	27	文员	3001	2500	20
3003	陈七	女	23	文员	3001	1800	20
4001	刘八	男	40	部门经理	1001	6500	30
4002	张九	男	35	业务员	4001	3500	30
4003	孙十	女	24	业务员	4001	2400	30
5001	沈一	男	38	部门经理	1001	6500	40
5002	吴二	女	32	程序员	5001	3500	40
5003	崔三	男	27	程序员	5001	3000	40
5004	刘四	男	19	程序员	5001	1800	40
6001	张五	女	35	部门经理	1001	4000	50
6002	韩六	男	26	维修员	6001	2200	50

图 8.9　员工表 emp 中的数据

（1）利用游标的简单循环，实现取出某一系别选修“大学英语”课程的学生姓名和成绩，并输出。（注：系别名称从键盘输入）

DEPTNO	DNAME	LOC
10	财务部	上海
20	人力资源部	北京
30	销售部	北京
40	研发部	上海
50	客服部	大连

图 8.10 部门表 dept 中的数据

(2) 利用游标的 FOR 循环,实现取出某一系别选修"JSP 程序设计"课程的学生姓名和成绩,并输出。(注:系别名称从键盘输入)

(3) 查询员工表 EMP 中"研发部"员工的基本情况,输出员工的姓名和工资;如果工资小于 3000 元,则将其工资改为 3000 元。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,使用显式游标的处理步骤,正确设定数据类型实现相应程序功能。程序无语法错误,书写规范,运行无误为优秀。如果出现错误,根据错误点数以及结构合理性灵活给分。

8.4.2 实验 2 带参数的游标

1. 实验目的

(1) 熟悉游标的概念和属性。

(2) 掌握游标的处理步骤和带参数游标的应用。

2. 实验内容

样本数据库中,学生表 student、课程表 course 和选课表 sc 的数据信息如图 8.11～图 8.13 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 8.11 学生表 student 中的数据

(1) 利用带参数的游标,实现从 student 表中查询"治安管理系"的学生学号和姓名,并输出。("治安管理系"为实参)

(2) 用带参数游标的 FOR 循环依此输出学生表中的每一个系别名称,在系别名称的下面输出该系别学生的姓名和年龄,结果按照年龄的降序排列。

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图 8.12　课程表 course 中的数据

SNO	CNO	GRADE
11432001	c1	75
11432001	c2	95
11432002	c1	82
11432002	c2	88
11432002	c4	76
11432002	c5	55
11432003	c1	65
11432003	c2	72
11432003	c3	90
11432003	c4	85
11432004	c1	93
11432004	c2	96
11432004	c5	91
11432005	c1	50
11432005	c2	70
11432005	c5	88
11432005	c8	43
11432007	c1	75
11432007	c2	73
11432007	c3	66
11432007	c6	82
11432007	c7	94
11432008	c1	82
11432008	c2	77
11432008	c3	85
11432008	c4	87
11432008	c5	82
11432008	c6	94
11432008	c7	92
11432008	c8	89
11432009	c1	86
11432009	c2	
11432009	c8	
11432010	c1	73
11432010	c2	
11432010	c8	76

图 8.13　选课表 sc 中的数据

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，使用显式游标的处理步骤，正确设定参数个数和数据类型实现相应程序功能。程序无语法错误，书写规范，运行无误为优秀。如果出现错误，根据错误点数以及结构合理性灵活给分。

8.4.3 实验3 隐式游标

1. 实验目的

(1) 熟悉隐式游标的概念和属性。

(2) 掌握隐式游标的应用。

2. 实验内容

样本数据库中,学生表student、课程表course和选课表sc的数据信息如图8.14~图8.16所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图8.14 学生表student中的数据

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图8.15 课程表course中的数据

(1) 随机输入一个学生学号,删除该学生的选课记录,如果删除操作成功,提示“删除成功”,并输出删除选课记录的行数,否则提示“删除失败”。

(2) 更新课程表,将课程编号为c9的课程名称改为Oracle,学分改为4。如果没有找到要更新的记录,则往课程表中插入该条记录,并输出提示信息“没有找到要更新的记录,插入新信息”。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,正确判定隐式游标的属性实现相应程序功能。程序无语法错误,书写规范,运行无误为优秀。如果出现错误,则根据错误点数以及结构合理性灵活给分。

```
SNO      CNO           GRADE
-------- -------- ----------
11432001 c1               75
11432001 c2               95
11432002 c1               82
11432002 c2               88
11432002 c4               76
11432002 c5               55
11432003 c1               65
11432003 c2               72
11432003 c3               90
11432003 c4               85
11432004 c1               93
11432004 c2               96
11432004 c5               91
11432005 c1               50
11432005 c2               70
11432005 c5               88
11432005 c8               43
11432007 c1               75
11432007 c2               73
11432007 c3               66
11432007 c6               82
11432007 c7               94
11432008 c1               82
11432008 c2               77
11432008 c3               85
11432008 c4               87
11432008 c5               82
11432008 c6               94
11432008 c7               92
11432008 c8               89
11432009 c1               86
11432009 c2
11432009 c8
11432010 c1               73
11432010 c2
11432010 c8               76
```

图 8.16　选课表 sc 中的数据

8.5　本章小结

本章首先介绍了游标的定义、作用和分类。游标充当指针的作用，使应用程序设计语言一次只能处理查询结果中的一行。

其次介绍了显式游标的处理步骤，显式游标的属性、游标的 FOR 循环、利用游标操纵数据库和带参数的游标。使用游标的 FOR 循环可以简化显式游标的处理步骤，它能够实现自动的打开游标、自动的循环取出当前行的结果提取到 PL/SQL 变量、自动的关闭游标三个步骤，它是一种快捷处理游标的方式。利用游标操纵数据库，在游标定义的时候，增加了 FOR UPDATE 从句；在游标使用的时候，增加了 WHERE CURRENT OF 子句。使用带参数的游标可以提高程序的灵活性，在定义游标的时候，增加了形参列表；在打开游标的时候，进行实参与形参值的传递。

最后介绍了隐式游标的属性、显式游标与隐式游标的区别。隐式游标的操作，如定义、

打开、取值及关闭操作，都由系统自动地完成，无须用户进行处理。用户只能通过隐式游标的相关属性，来完成相应的操作。

8.6 课后习题

一、选择题

1. 游标在处理的过程中需要分四步来处理，其中哪一步需要循环？（ ）

A. 定义游标　　B. 打开游标

C. 从游标取值到变量　　D. 关闭游标

2. 定义游标时定义了游标的名字，并将该游标和一个 SELECT 语句相关联。这个 SELECT 语句中不可能出现的语句是（ ）。

A. WHERE　　B. ORDER BY　　C. INTO　　D. GROUP BY

3. 游标的四个属性中，哪一个属性的取值与其他三个属性的取值类型不同？（ ）

A. 游标名%NOTFOUND　　B. 游标名%FOUND

C. 游标名%ROWCOUNT　　D. 游标名%ISOPEN

4. 对于游标 FOR 循环，以下哪一种说法是不正确的？（ ）

A. 循环隐含使用 FETCH 获取数据

B. 循环隐含使用 OPEN 打开记录集

C. 终止循环操作也就关闭了游标

D. 游标 FOR 循环不需要定义游标

5. 通过游标操纵数据库，以下哪一种说法是不正确的？（ ）

A. 在定义游标的查询语句时，必须加上 FOR UPDATE 从句

B. 使用 FOR UPDATE OF 从句表示对表加锁，OF 列不可以省略

C. 当用户从一张表或多张表中查询多条记录时，必须使用一个显式游标

D. CURRENT OF 从句表示允许用户对 FETCH 语句取出的当前行进行更新和删除

二、简答题

1. 简述显式游标的使用步骤。

2. 简述显式游标和隐式游标的区别。

三、应用题

1. 从学生表中取出年龄在 18～20 岁的学生姓名和性别，并将其输出。

2. 用带参数的游标，实现输出“财务部”员工的编号和姓名。（注：“财务部”为实参）

第 9 章

存储过程

前几章介绍了 PL/SQL 无名块。PL/SQL 无名块不存储在数据库中，并且不能被其他的 PL/SQL 程序块调用。而 Oracle 可以将 PL/SQL 程序块存储在数据库中，并可以在任何地方来运行它，这种 PL/SQL 程序块被称为 PL/SQL 子程序，它们是被命名的 PL/SQL 程序块，可以在客户端与服务器端的任何工具和任何应用中运行。

存储子程序是以独立对象的形式存储在数据库服务器中的，主要包括存储过程和存储函数两种，本章主要介绍存储过程的使用。

9.1 存储过程的创建

存储过程是一种命名的 PL/SQL 程序块，它可以被赋予参数，存储在数据库中，可以被用户调用。由于存储过程是已经编译好的代码，所以在调用的时候不必再次进行编译，从而提高了程序的运行效率。另外，使用存储过程可以实现程序的模块化设计。

9.1.1 创建过程的语法

创建存储过程的基本语法：

```
CREATE [OR REPLACE] PROCEDURE  过程名
       [(参数名 [IN | OUT | IN OUT] 数据类型, …)]
{IS | AS}
       [说明部分]
BEGIN
       语句序列
       [EXCEPTION 出错处理]
END [过程名];
```

说明：

(1) 过程名和参数名必须符合 Oracle 中标识符命名规则。

(2) OR REPLACE 是一个可选的关键字，建议用户使用此关键字，当数据库中已经存在此过程名，则该过程会被重新定义，并被替换。

(3) 关键字 IS 和 AS 本身没有区别，选择其中一个即可。

(4) IS 后面是一个完整的 PL/SQL 程序块的三个部分，可以定义局部变量、游标等，但不能以 DECLARE 开始。

9.1.2 形式参数的三种类型

创建存储过程时，可以定义零个或多个形式参数。形式参数主要有三种模式，包括 IN、OUT、IN OUT。如果定义形参时没有指定参数的模式，那么系统默认该参数默认模式为 IN 模式。

在声明形参时，不能定义形参的长度或精度，它们是作为参数传递机制的一部分被传递的，是由实参决定的。可以使用%TYPE 或%ROWTYPE 定义形参，%TYPE 或%ROWTYPE 只是隐含地包括长度或精度等约束信息。

三种模式参数的具体描述如表 9.1 所示。

表 9.1 三种模式参数的具体描述

模 式	描 述
IN(默认模式)参数	输入参数，用来从调用环境中向存储过程传递值，在过程体内不能给 IN 参数赋值
OUT 参数	输出参数，用来从存储过程中返回值给调用者，在过程体内必须给 OUT 参数赋值
IN OUT 参数	输入输出参数，既可以从调用者向存储过程中传递值，也可以从过程中返回可能改变的值给调用者

例题 9.1 创建一个无参数的存储过程，输出当前系统的日期。

例题解析：

```
CREATE OR REPLACE PROCEDURE out_date
IS
BEGIN
DBMS_OUTPUT.PUT_LINE('当前系统日期为：'||SYSDATE);
END out_date;
```

例题 9.2 创建一个带输入参数的存储过程，给某一指定的员工涨指定数量的工资。

例题解析：

```
CREATE OR REPLACE PROCEDURE raise_salary
    (v_empno IN emp.empno%TYPE,v_salary number)
IS
BEGIN
    UPDATE emp SET salary = salary + v_salary WHERE empno = v_empno;
END raise_salary;
```

例题 9.3 创建一个带输入和输出参数的存储过程，根据给定的学生学号返回该学生

的姓名和系别名称。

例题解析：

```
CREATE OR REPLACE PROCEDURE query_student
    (v_sno IN student.sno%TYPE,
    v_sname OUT student.sname%TYPE,
    v_dept OUT student.dept%TYPE)
IS
BEGIN
    SELECT sname,dept INTO v_sname,v_dept FROM student WHERE sno = v_sno;
END query_student;
```

9.2　存储过程的调用

存储过程创建后，以编译的形式存储于数据库的数据字典中。如果不被调用，存储过程是不会执行的。

9.2.1　参数传值

通过存储过程的名称调用存储过程时，实参的数量、顺序、类型要与形参的数量、顺序、类型相匹配。

如果形式参数是 IN 模式的参数，实际参数可以是一个具体的值，或是一个已经赋值的变量。

如果形式参数是 OUT 模式的参数，实际参数必须是一个变量，而不能是常量。当调用存储过程后，此变量就被赋值了。

如果形式参数是 IN OUT 模式的参数，则实际参数必须是一个已经赋值的变量。当存储过程完成后，该变量将被重新赋值。

9.2.2　调用方法

(1) 在 SQL*Plus 中调用存储过程

在 SQL*Plus 中可以使用 EXECUTE 命令调用存储过程。

(2) 在 PL/SQL 程序中调用存储过程

在 PL/SQL 程序中，存储过程可以作为一个独立的表达式被调用。

例题 9.4　利用两种不同的方法调用例题 9.1 中的存储过程 out_date，查询当前系统日期。程序运行效果如图 9.1 所示。

例题解析：

程序一：

```
EXECUTE out_date;
```

```
当前系统日期为：17-10月-14
PL/SQL 过程已成功完成。
```

图 9.1 调用存储过程 out_date 输出当前系统日期

程序二：

```
BEGIN
    out_date;
END;
```

例题 9.5 从 PL/SQL 程序中调用例题 9.2 中的存储过程 raise_salary，从键盘随机输入员工编号和涨薪额度，实现对该员工涨指定数量的工资。程序运行效果如图 9.2 所示。

```
输入 p_empno 的值:  2003
原值    2:     v_empno emp.empno%TYPE:=&p_empno;
新值    2:     v_empno emp.empno%TYPE:=2003;
输入 p_salary 的值:  800
原值    3:     v_salary emp.salary%TYPE:=&p_salary;
新值    3:     v_salary emp.salary%TYPE:=800;

PL/SQL 过程已成功完成。
```

图 9.2 调用存储过程 raise_salary 给编号 2003 员工的工资涨 800 元

例题解析：

```
DECLARE
    v_empno emp.empno%TYPE: = &p_empno;
    v_salary emp.salary%TYPE: = &p_salary;
BEGIN
     raise_salary(v_empno,v_salary);
END;
```

例题 9.6 从 PL/SQL 程序中调用例题 9.3 中的存储过程 query_student，实现查询学号为 11432003 的学生姓名和系别名称，并输出。程序运行效果如图 9.3 所示。

```
学生姓名为:张三,系别名称为:侦查系
PL/SQL 过程已成功完成。
```

图 9.3 调用存储过程 query_student 查询学生信息

例题解析：

```
DECLARE
    v_sname student.sname%TYPE;
    v_dept student.dept%TYPE;
BEGIN
     query_student('11432003',v_sname,v_dept);
```

```
    DBMS_OUTPUT.PUT_LINE('学生姓名为:'||v_sname||','||'系别名称为:'|| v_dept);
END;
```

9.3　存储过程的管理

9.3.1　修改存储过程

为了修改存储过程,可以先删除该存储过程,然后重新创建。也可以采用 CREATE OR REPLACE PROCEDURE 语句重新创建并覆盖原有的存储过程。

例题 9.7　修改例题 9.2 中的存储过程 raise_salary,实现给某一指定的员工减少指定数量的工资。

例题解析:

```
CREATE OR REPLACE PROCEDURE raise_salary
(v_empno IN emp.empno%TYPE,v_salary number)
IS
BEGIN
UPDATE emp SET salary = salary - v_salary WHERE empno = v_empno;
END raise_salary;
```

9.3.2　删除存储过程

删除存储过程使用 DROP PROCEDURE 语句。

例题 9.8　删除例题 9.2 中的存储过程 raise_salary。

例题解析:

```
DROP PROCEDURE raise_salary;
```

9.3.3　查看语法错误

存储过程在编译时可能出现一些语法错误,但只是以警告的方式提示"创建的过程带有编译错误",用户如果想查看错误的详细信息,可以使用 SHOW ERRORS 命令显示刚编译的存储过程的出错信息。

例题 9.9　根据图 9.4 给出的存储过程的定义,查看它的语法错误。

例题解析:

```
SHOW ERRORS;
```

查看存储过程的语法错误,运行效果如图 9.5 所示。

```
SQL> CREATE OR REPLACE PROCEDURE p_emp
  2  (v_empno IN emp.empno%TYPE,
  3  v_ename OUT char(8),
  4  v_sal  OUT emp.sal%TYPE)
  5  IS
  6  BEGIN
  7  SELECT ename,sal INTO v_ename,v_sal FROM emp
  8  WHERE empno=v_empno;
  9  /

警告: 创建的过程带有编译错误。
```

图 9.4　创建存储过程 p_emp 的程序运行效果

```
PROCEDURE P_EMP 出现错误:

LINE/COL ERROR
-------- -----------------------------------------------------------------
3/17     PLS-00103: 出现符号 "("在需要下列之一时:
         := ) , default varying
         character large
         符号 ":=" 被替换为 "(" 后继续。

8/20     PLS-00103: 出现符号 "end-of-file"在需要下列之一时:
         begin case declare
         end exception exit for goto if loop mod null pragma raise
         return select update while with <an identifier>
         <a double-quoted delimited-identifier> <a bind variable> <<
         close current delete fetch lock insert open rollback
         savepoint set sql execute commit forall merge pipe
```

图 9.5　查看语法错误的程序运行效果

9.3.4　查看结构

查看存储过程的基本结构,包括存储过程的形式参数名称、形式参数的模式以及形式参数的数据类型可以通过执行 DESC 命令获得。

例题 9.10　查看存储过程 query_student 的基本结构。

例题解析:

```
DESC query_student;
```

程序运行效果如图 9.6 所示。

```
PROCEDURE query_student
参数名称                        类型                    输入/输出默认值?
------------------------------ ----------------------- ------ --------
 V_SNO                          CHAR(8)                 IN
 V_SNAME                        VARCHAR2(20)            OUT
 V_DEPT                         VARCHAR2(20)            OUT
```

图 9.6　存储过程 query_student 的基本结构

9.3.5　查看源代码

存储过程的源代码通过查询数据字典 USER_SOURCE 中的 TEXT 即可获得。在数据字典中,存储过程的名字是以大写方式存储的。

```
SELECT TEXT
FROM USER_SOURCE
WHERE NAME = '过程名';
```

例题 9.11　查看存储过程 query_student 的源代码。

例题解析:

```
SELECT TEXT
FROM USER_SOURCE
WHERE NAME = 'QUERY_STUDENT';
```

程序运行效果如图 9.7 所示。

```
TEXT
--------------------------------------------------------------------
PROCEDURE query_student
(v_sno IN student.sno%TYPE,
v_sname OUT student.sname%TYPE,
v_dept OUT student.dept%TYPE)
IS
BEGIN
SELECT sname,dept INTO v_sname,v_dept FROM student WHERE sno =v_sno;
END query_student;

已选择8行。
```

图 9.7　存储过程 query_student 的源代码

9.4　实　　验

9.4.1　实验 1　不带参数的存储过程

1. 存储过程

(1) 掌握不带参数存储过程的创建方法。

(2) 掌握不带参数存储过程的调用方法。

2. 实验内容

样本数据库中,学生表 student 的数据信息如图 9.8 所示。

(1) 创建一个统计全校学生总人数的无参数存储过程。

(2) 调用此过程:显示当前学校总人数。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 9.8　学生表 student 中的数据

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，创建存储过程实现相应程序功能。程序无语法错误，书写规范，运行无误为优秀。如果出现错误，根据错误点数以及结构合理性灵活给分。

9.4.2 实验2 带参数的存储过程

1. 存储过程

(1) 掌握带参数存储过程的创建方法。

(2) 掌握带参数存储过程的调用方法。

2. 实验内容

在样本数据库中，学生表 student、课程表 course 和选课表 sc 的数据信息如图 9.9～图 9.11 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 9.9　学生表 student 中的数据

(1) 练习带 in 型参数的存储过程的创建与调用。

① 创建一个带输入参数的存储过程：根据给定的学生学号，输出该学生的姓名和年龄。

② 调用此存储过程：输出学号为 11432001 的学生的姓名和年龄。

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图 9.10　课程表 course 中的数据

SNO	CNO	GRADE
11432001	c1	75
11432001	c2	95
11432002	c1	82
11432002	c2	88
11432002	c4	76
11432002	c5	55
11432003	c1	65
11432003	c2	72
11432003	c3	90
11432003	c4	85
11432004	c1	93
11432004	c2	96
11432004	c5	91
11432005	c1	50
11432005	c2	70
11432005	c5	88
11432005	c8	43
11432007	c1	75
11432007	c2	73
11432007	c3	66
11432007	c6	82
11432007	c7	94
11432008	c1	82
11432008	c2	77
11432008	c3	85
11432008	c4	87
11432008	c5	82
11432008	c6	94
11432008	c7	92
11432008	c8	89
11432009	c1	86
11432009	c2	
11432009	c8	
11432010	c1	73
11432010	c2	
11432010	c8	76

图 9.11　选课表 sc 中的数据

(2) 练习带 out 型参数的存储过程的创建与调用。

① 创建一个带输入参数的存储过程：根据给定的系别名称，输出该系别所有学生的学号和姓名。

② 调用此存储过程：输出“公安信息系”的学生学号和姓名。

(3) 练习带 in 与 out 型参数的存储过程的创建与调用。

① 创建一个带输入和输出参数的存储过程：根据给定的学生学号，返回该学生的平均成绩。

② 调用此存储过程：输出学号为 11432002 的学生的平均成绩。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，创建存储过程实现相应程序功能。参数的输入和输出定义准确，程序无语法错误，书写规范，运行无误为优秀。如果出现错误，则根据错误点数以及结构合理性灵活给分。

9.5 本章小结

存储子程序是被命名的 PL/SQL 块，是 PL/SQL 程序模块化的一种体现。PL/SQL 中的存储子程序包括存储过程和存储函数两种。

本章首先介绍了存储过程创建的语法格式，并详细讲述了三种类型的形式参数。其次介绍了存储过程的两种调用方法。最后介绍了存储过程的管理命令，包括如何修改存储过程、删除存储过程、查看存储过程的语法错误、查看存储过程的结构及查看源代码。

9.6 课后习题

一、选择题

1. 存储过程中可以使用参数，IN 模式参数为（　　）。

   A. 输入参数　　B. 输出参数　　C. 输入输出参数　　D. 只读参数

2. 下列哪个语句可以在 SQL*Plus 中直接调用一个过程？（　　）

   A. RETURN　　B. EXECUTE　　C. CALL　　D. SET

3. 下列有关存储过程的特点说法错误的是（　　）。

   A. 存储过程不能将值传回调用的主程序

   B. 存储过程是一个命名的模块

   C. 编译的存储过程存放在数据库中

   D. 一个存储过程可调用另一个存储过程

二、应用题

1. 创建并调用存储过程，完成下列功能：

（1）创建一个带输入和输出参数的存储过程：根据给定的员工编号，返回该员工的姓名和工资。

（2）在 PL/SQL 块中调用此过程，输出编号为 2001 的员工的姓名和工资。

2. 创建并调用存储过程,完成下列功能:

(1) 创建一个存储过程,根据员工编号(empno),更改该员工的姓名,如果该员工不存在,就触发用户自定义的异常,在屏幕上显示"不存在该员工"。

(2) 在 PL/SQL 块中调用此过程,将 1001 号员工的姓名改为"天天"。

3. 创建并调用存储过程,完成下列功能:

(1) 创建一个存储过程:根据某个学生编号输出该学生选修课程的编号和成绩。

(2) 在 PL/SQL 块中调用此过程,输出学号为 11432002 的学生选修的课程号和成绩。

第 10 章

存储函数

10.1 存储函数的创建

存储函数的创建与存储过程的创建基本相似，不同的地方是存储函数必须有返回值。

10.1.1 创建函数的语法

创建存储函数的基本语法：

```
CREATE [OR REPLACE] FUNCTION 函数名
        [(参数名 [IN] 数据类型, …)]
RETURN 数据类型
{IS | AS}
        [说明部分]
BEGIN
        语句序列
        RETURN(表达式)
        [EXCEPTION
        例外处理程序]
END [函数名];
```

说明：

(1) 函数名和参数名必须符合 Oracle 中标识符命名规则。

(2) OR REPLACE 是一个可选的关键字，建议用户使用此关键字，当数据库中已经存在此过程名，则该过程会被重新定义，并被替换。

(3) 关键字 IS 和 AS 本身没有区别，选择其中一个即可。

(4) IS 后面是一个完整的 PL/SQL 程序块的三个部分，可以定义局部变量、游标等，但不能以 DECLARE 开始。

10.1.2 形式参数与返回值

与存储过程相似，创建存储函数时，可以定义零个或多个形式参数，并且都为 IN 模式，IN 可以省略不写。存储函数是靠 RETURN 语句返回结果，并且只能返回一个结果。在函

数定义的头部，参数列表之后，必须包含一个 RETURN 语句来指明函数返回值的类型，但不能约束返回值的长度、精度等。

在函数体的定义中，必须至少包含一个 RETURN 语句，来指明函数的返回值。也可以有多个 RETURN 语句，但最终只有一个 RETURN 语句被执行。

例题 10.1　创建一个无参数的存储函数，返回员工表中员工的最高工资。

例题解析：

```
CREATE OR REPLACE FUNCTION max_salary
RETURN emp.salary%TYPE
IS
    v_salary emp.salary%TYPE;
BEGIN
    SELECT MAX(salary) INTO v_salary FROM emp;
    RETURN v_salary;
END max_salary;
```

例题 10.2　创建一个有参数的存储函数，根据给定的系别名称，返回该系别的学生人数。

例题解析：

```
CREATE OR REPLACE FUNCTION num_dept
    (v_dept student.dept%TYPE)
RETURN number
IS
    v_num NUMBER;
BEGIN
    SELECT COUNT( * ) INTO v_num FROM student WHERE dept = v_dept;
    RETURN v_num;
END num_dept;
```

10.2　存储函数的调用

存储函数创建以后，可以使用以下两种方法调用存储函数。

(1) 在 SQL 语句中调用存储函数。

(2) 在 PL/SQL 程序中调用存储函数。

调用存储函数与调用存储过程不同，调用函数时，需要一个变量来保存返回的结果值，这样函数就组成了表达式的一部分。

例题 10.3　利用两种不同的方法调用例题 10.1 中的存储函数 max_salary，查询员工的最高工资。

例题解析：

(1) 在 SQL 语句中调用存储函数。

```
SELECT max_salary FROM dual;
```

程序运行效果如图 10.1 所示。

```
MAX_SALARY
----------
      9500
```

图 10.1 在 SQL 语句中调用存储函数 max_salary 的程序运行效果

(2) 在 PL/SQL 程序中调用存储函数。

```
DECLARE
    v_salary emp.salary%TYPE;
BEGIN
    v_salary: = max_salary;
    DBMS_OUTPUT.PUT_LINE('员工的最高工资为:'||v_salary);
END;
```

程序运行效果如图 10.2 所示。

```
员工的最高工资为:9500
PL/SQL 过程已成功完成。
```

图 10.2 在 PL/SQL 程序中调用存储函数 max_salary 的程序运行效果

例题 10.4 从 PL/SQL 程序中调用例题 10.2 中的存储函数 num_dept,从键盘随机输入系别名称,返回该系别的学生人数,并输出。

例题解析:

```
DECLARE
    v_dept student.dept%TYPE: = &p_dept;
    v_number number;
BEGIN
    v_number: = num_dept(v_dept);
    DBMS_OUTPUT.PUT_LINE('该系别的学生人数为:'||v_number);
END;
```

程序运行效果如图 10.3 所示。

```
输入 p_dept 的值:  '侦查系'
原值    2: v_dept student.dept%TYPE:=&p_dept;
新值    2: v_dept student.dept%TYPE:='侦查系';
该系别的学生人数为:3

PL/SQL 过程已成功完成。
```

图 10.3 在 PL/SQL 程序中调用存储函数 num_dept 的程序运行效果

10.3　存储函数的管理

10.3.1　修改存储函数

可以使用 CREATE OR REPLACE FUNCTION 语句重新创建并覆盖原有的存储函数。

例题 10.5　修改例题 10.1 中的存储函数 max_salary，查询编号为 10 部门的员工最高工资。

例题解析：

```
CREATE OR REPLACE PROCEDURE raise_salary
(v_empno IN emp.empno%TYPE, v_salary number)
IS
BEGIN
UPDATE emp SET salary = salary - v_salary WHERE deptno = '10' AND empno = v_empno;
END raise_salary;
```

10.3.2　删除存储函数

删除存储函数使用 DROP FUNCTION 语句。

例题 10.6　删除例题 10.1 中的存储函数 max_salary。

例题解析：

```
DROP FUNCTION max_salary;
```

10.3.3　查看语法错误

查看刚编译的存储函数出现错误的详细信息，使用 SHOW ERRORS 命令。

例题 10.7　根据图 10.4 给出的存储函数的定义，查看它的语法错误。

```
SQL> CREATE OR REPLACE FUNCTION avg_grade
(v_sno in char(8))
return number
is
DECLARE
v_avg  number;
begin
select avg(grade) into v_avg from sc where sno=v_sno;
end;
/

警告: 创建的函数带有编译错误。
```

图 10.4　创建存储函数 avg_grade 的程序运行效果

例题解析：

```
SHOW ERRORS;
```

查看存储函数的语法错误，运行效果如图10.5所示。

```
LINE/COL ERROR
-------- ----------------------------------------------------------------
2/15     PLS-00103: 出现符号 "("在需要下列之一时：
         := ) , default varying
         character large
         符号 ":=" 被替换为 "(" 后继续。

5/1      PLS-00103: 出现符号 "DECLARE"在需要下列之一时：
         begin function package
         pragma procedure subtype type use <an identifier>
         <a double-quoted delimited-identifier> form current cursor
         external language
         符号 "begin" 被替换为 "DECLARE" 后继续。

9/4      PLS-00103: 出现符号 "end-of-file"在需要下列之一时：
         begin case declare
         end exception exit for goto if loop mod null pragma raise
         return select update while with <an identifier>
         <a double-quoted delimited-identifier> <a bind variable> <<
         close current delete fetch lock insert open rollback
         savepoint set sql execute commit forall merge pipe
```

图10.5　查看语法错误的程序运行效果

10.3.4　查看结构

查看存储函数的基本结构，包括存储函数的形式参数名称、形式参数的数据类型以及返回值类型可以通过执行DESC命令获得。

例题10.8　查看存储函数num_dept的基本结构。

例题解析：

```
DESC num_dept;
```

程序运行效果如图10.6所示。



```
FUNCTION num_dept RETURNS NUMBER
参数名称                          类型                     输入/输出默认值?
-------------------------------- ----------------------- ------ --------
 V_DEPT                           VARCHAR2(20)             IN
```

图10.6　存储函数num_dept的基本结构

10.3.5　查看源代码

存储函数的源代码通过查询数据字典USER_SOURCE中的TEXT即可获得。

```
SELECT TEXT
FROM USER_SOURCE
WHERE NAME = '函数名';
```

例题 10.9　查看存储函数 num_dept 的源代码。

例题解析：

```
SELECT TEXT
FROM USER_SOURCE
WHERE NAME = 'NUM_DEPT';
```

程序运行效果如图 10.7 所示。

```
TEXT
------------------------------------------------------------
FUNCTION num_dept
(v_dept student.dept%TYPE)
RETURN number
IS
v_num NUMBER;
BEGIN
SELECT COUNT(*) INTO v_num FROM student WHERE dept=v_dept;
RETURN v_num;
END num_dept;

已选择9行。
```

图 10.7　存储函数 num_dept 的源代码

10.4　存储过程与存储函数的区别

存储过程和存储函数的差别主要有两个：一是返回值的方法不同，二是调用方法不同。

10.4.1　返回值方法不同

存储过程：有零个或多个参数，过程不返回值，其返回值是靠 OUT 参数带出来的。

存储函数：有零个或多个参数，但不能有 OUT 参数。函数只返回一个值，靠 RETURN 子句返回。

10.4.2　调用方法不同

存储过程：调用存储过程的语句可以作为独立的可执行语句在 PL/SQL 程序块中单独出现。例如：

```
过程名(实际参数 1,实际参数 2...);
```

存储函数：函数可以在任何表达式能够出现的地方被调用，调用函数的语句不能作为可执行语句单独出现在 PL/SQL 程序块中。例如：

```
变量名:= 函数名(实际参数 1,实际参数 2...);
```

10.5 实　　验

10.5.1 实验1 不带参数的存储函数

1. 实验目的

(1) 掌握不带参数的存储函数的创建方法。

(2) 掌握不带参数的存储函数的调用方法。

(3) 理解存储过程与存储函数的区别。

2. 实验内容

样本数据库中,学生表student的数据信息如图10.8所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图10.8　学生表student中的数据

(1) 创建一个返回学生总人数的无参数函数。

(2) 调用此存储函数:输出当前学生的总人数。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求创建存储函数,实现相应程序功能。返回值的类型定义准确,程序无语法错误,书写规范,运行无误为优秀。如果出现错误,则根据错误点数以及结构合理性灵活给分。

10.5.2 实验2 带参数的存储函数

1. 实验目的

(1) 掌握带参数的存储函数的创建方法。

(2) 掌握带参数的存储函数的调用方法。

(3) 理解存储过程与存储函数的区别。

2. 实验内容

样本数据库中，学生表 student、课程表 course 和选课表 sc 的数据信息如图 10.9～图 10.11 所示。

（1）练习带 in 型参数的存储函数的创建与调用。

① 创建存储函数：根据给定的学号，返回该学生的总成绩。

② 调用此存储函数：输出学号为 11432002 的学生的总成绩。

（2）练习带 out 型参数的存储函数的创建与调用。

① 创建存储函数：根据给定的学生学号，返回该学生的详细信息。

② 调用此存储函数：输出学号为 11432003 的学生姓名、性别和年龄。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求创建存储函数，实现相应程序功能。参数的输入定义和返回值的类型定义准确，程序无语法错误，书写规范，运行无误为优秀。如果出现错误，则根据错误点数以及结构合理性灵活给分。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 10.9　学生表 student 中的数据

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图 10.10　课程表 course 中的数据

SNO	CNO	GRADE
11432001	c1	75
11432001	c2	95
11432002	c1	82
11432002	c2	88
11432002	c4	76
11432002	c5	55
11432003	c1	65
11432003	c2	72
11432003	c3	90
11432003	c4	85
11432004	c1	93
11432004	c2	96
11432004	c5	91
11432005	c1	50
11432005	c2	70
11432005	c5	88
11432005	c8	43
11432007	c1	75
11432007	c2	73
11432007	c3	66
11432007	c6	82
11432007	c7	94
11432008	c1	82
11432008	c2	77
11432008	c3	85
11432008	c4	87
11432008	c5	82
11432008	c6	94
11432008	c7	92
11432008	c8	89
11432009	c1	86
11432009	c2	
11432009	c8	
11432010	c1	73
11432010	c2	
11432010	c8	76

图 10.11　选课表 sc 中的数据

10.6　本 章 小 结

本章首先介绍了存储函数创建的语法格式，给出了形式参数类型和函数返回值 RETURN 子句。介绍了存储函数的两种调用方法：一种是在 SQL 语句中调用存储函数，另一种是在 PL/SQL 程序中调用存储函数。

其次介绍了存储函数的管理命令，包括存储函数的修改、存储函数的删除、存储函数语法错误的查看、存储函数结构的查看、存储函数源代码的查看等。

最后介绍了存储函数与存储过程的区别，分别从返回值方法和调用方法两个方面进行了阐述。

10.7　课 后 习 题

一、选择题

1. 存储函数的关键字是(　　)。

A. PROCEDURE　B. FUNCTION　C. PACKAGE　D. TRIGGER

2. 关于存储过程和函数的区别,下列说法错误的是(　　)。

A. 存储过程可以有多个 out 参数,函数只能有一个返回值

B. 存储过程可以单独调用,函数的语句不能作为可执行语句单独出现在 PL/SQL 块中

C. 用 EXECUTE 既可以调用过程,也可以调用函数

D. 函数必须要有返回数据类型

3. 对于存储函数的参数和返回值描述不正确的是(　　)。

A. 存储函数的形式参数只能是 in 模式

B. 存储函数有零个或多个 in 型参数

C. 存储函数的返回值使用 out 型参数返回

D. 存储函数的返回值使用 return 子句返回

4. 下列有关存储函数的特点说法不正确的是(　　)。

A. 存储函数是一个命名的程序块

B. 存储函数不能将值传回到调用它的主程序

C. 编译后的存储函数存放在数据库的数据字典中

D. 一个存储函数可以调用另一个存储函数

二、应用题

1. 创建存储函数,实现下列功能。

(1) 创建一个存储函数。有一个输入参数员工编号,根据该员工编号,返回该员工的所有信息。

(2) 在 PL/SQL 块中调用此函数,输出 3002 号员工的姓名和工资。

2. 创建存储函数,实现下列功能。

(1) 创建一个存储函数:根据指定的学生编号返回该学生的平均成绩。

(2) 在 PL/SQL 块中调用此函数,输出学号为 11432001 学生的平均成绩。

第 11 章

包

11.1 包的简介

11.1.1 包的定义

PL/SQL 程序包是将一组相关过程、函数、变量、常量和游标等 PL/SQL 程序设计元素组织在一起，成为一个完整的单元，编译后存储在数据库的数据字典中，作为一种全局结构，供应用程序调用。与存储过程和函数一样，包被存放在数据库的数据字典中。包中可以包含的元素有存储过程、存储函数、游标、变量、常量、异常、记录类型等。

在 Oracle 数据库中，包有两类：一类是 Oracle 内置包，每个包是实现特定应用的过程、函数、常量等的集合；另一类是根据应用需要由用户创建的包。

11.1.2 包的优点

当完成一项复杂的工作任务时，可能需要用到多个存储过程或者函数，这时可以考虑把这些存储过程或者函数集成到一个包中，从而方便对于存储过程和函数的管理。包的优点如下：

(1) 加强了程序开发的模块化。

(2) 可以隐藏信息，提高数据的安全性。

(3) 为用户提供全局的信息。

(4) 当第一次调用包时，包被整体读入内存，在后续的调用中不会耗费额外的 I/O，提高了程序执行的性能。

11.2 包的创建与调用

包的创建分为两个步骤，分别为包说明(PACKAGE)的创建和包主体(PACKAGE BODY)的创建，包说明和包主体分开编译，并作为两个分开的对象存放在数据库字典中。

11.2.1　包说明的创建

包说明部分声明包中所有可共享的元素，包括数据类型、变量、常量、游标、过程、函数和异常错误处理等元素，这些元素被称为包的公有元素。在包说明中声明的元素不仅可以在包的内部使用，也可以被应用程序调用。

包说明创建的语法如下：

```
CREATE [OR REPLACE] PACKAGE 包名
{IS | AS}
        公共变量的定义
        公共类型的定义
        公共出错处理的定义
        公共游标的定义
        函数说明
        过程说明
END[包名];
```

说明：

(1) 元素声明的顺序可以是任意的，但必须先声明后使用。

(2) 所有元素都是可选的。

(3) 过程和函数的声明只包括原型信息，不包括任何实现代码。

11.2.2　包主体的创建

包主体是包说明部分的具体实现，它包含了包说明部分所声明的过程和函数的实现代码。此外，包主体中还可以包含在包说明中没有声明的数据类型、变量、常量、游标、过程、函数和异常错误处理等元素，这些元素被称为包的私有元素，只能由同一个包中的过程或函数使用。

1. 包元素的性质

包中的元素也分为公有元素和私有元素两种，这两种元素的区别是它们的作用域不同。公有元素不仅可以被包中的函数、过程调用，也可以被包外的 PL/SQL 程序访问，而私有元素只能被包内的函数和过程访问。

包元素的性质及描述如表 11.1 所示。

表 11.1　包元素的性质

元素的性质	描　　述	在包中的位置
公共的	在整个应用的全过程均有效	在包说明部分说明，并在包主体中具体定义
私有的	对包以外的存储过程和函数是不可见的	在包主体部分说明和定义
局部的	只在一个过程或函数内部可用	在所属过程或函数的内部说明和定义

2. 包主体创建的语法

```
CREATE [OR REPLACE] PACKAGE BODY 包名
{IS | AS}
私有变量的定义
   私有类型的定义
    私有出错处理的定义
    私有游标的定义
    函数定义
    过程定义
END[包名];
```

说明：

（1）包主体中过程和函数的原型必须与包说明中的声明完全一致；

（2）只有在包说明已经创建的前提下，才可以创建包主体；

（3）如果包声明中不包含任何过程和函数，则可以不创建包主体。

11.2.3 包的调用

在包说明中声明的任何元素都是公有的，在包的外部都是可见的，可以通过"包名.元素名"的形式进行调用，在包主体中可以通过"元素名"直接进行调用。但是，在包主体中定义而没有在包说明中声明的元素是私有的，只能在包主体中被引用。

包中的存储过程与存储函数的调用方法和前面讲的单独的存储过程与存储函数的调用方法基本相同，唯一的区别在于在被调用的存储过程和存储函数前必须指明其所在包的名字。

例题 11.1 创建一个包，包名为 stu_package。其中包括一个存储过程，用于根据学生学号返回该学生的选课数量；还包括一个存储函数，用于根据学生学号返回该学生的平均成绩。

例题解析：

（1）包说明的创建。

```
CREATE OR REPLACE PACKAGE stu_package
IS
    PROCEDURE get_num (v_sno IN student.sno%TYPE,v_num OUT NUMBER);
    FUNCTION get_grade (v_sno student.sno%TYPE)
    RETURN sc.grade%TYPE;
END stu_package;
```

（2）包主体的创建。

```
CREATE OR REPLACE PACKAGE BODY stu_package
IS
    PROCEDURE get_num(v_sno IN student.sno%TYPE,v_num OUT NUMBER)
```

```
    IS
    BEGIN
        SELECT COUNT(CNO) INTO v_num FROM sc WHERE sno = v_sno;
END get_num;
FUNCTION get_grade(v_sno student.sno%TYPE)
RETURN sc.grade%TYPE
IS
    v_grade sc.grade%TYPE;
BEGIN
    SELECT AVG(grade) INTO v_grade FROM sc WHERE sno = v_sno;
    RETURN v_grade;
    END get_grade;
END stu_package;
```

例题 11.2 　从 PL/SQL 程序中调用包 stu_package 中的存储过程 get_num，查询学号为 11432003 的学生的选课数量，并输出。

例题解析：

```
DECLARE
    v_num NUMBER;
BEGIN
     stu_package.get_num('11432003',v_num);
     DBMS_OUTPUT.PUT_LINE('该学生的选课数量为:'||v_num);
END;
```

程序运行效果如图 11.1 所示。

```
该学生的选课数量为:4
PL/SQL 过程已成功完成。
```

图 11.1　在包 stu_package 中调用存储过程 get_num 的程序运行效果

例题 11.3 　从 PL/SQL 程序中调用包 stu_package 中的存储函数 get_grade，返回学号为 11432003 的学生的平均成绩，并输出。

例题解析：

```
DECLARE
    v_grade sc.grade%TYPE;
BEGIN
    v_grade: = stu_package.get_grade('11432003');
    DBMS_OUTPUT.PUT_LINE('该学生的平均成绩为:'||v_grade);
END;
```

程序运行效果如图 11.2 所示。

```
该学生的平均成绩为:78
PL/SQL 过程已成功完成。
```

图 11.2 在包 stu_package 中调用存储函数 get_grade 的程序运行效果

11.3 包 的 重 载

在包的内部，过程和函数可以被重载，但需要注意以下几点：

(1) 重载子程序必须同名不同参，即名称相同，参数不同。参数的不同体现在参数的个数、顺序、类型等不同。

(2) 如果两个子程序参数仅是名称和模式不同，则这两个子程序不能重载。

例如，以下两个过程不能进行重载。

```
PROCEDURE overloading(parameter1 IN NUMBER);
PROCEDURE overloading(parameter2 OUT NUMBER);
```

(3) 不能仅根据两个函数返回值类型不同而对它们进行重载。

例如，以下两个函数不能进行重载。

```
FUNCTION overloading RETURN CHAR;
FUNCTION overloading RETURN DATE;
```

(4) 重载子程序的参数的类型系列方面必须不同。

例如，下面的重载是错误的。

```
PROCEDURE overloading(parameter1 IN CHAR);
PROCEDURE overloading(parameter2 IN VARCHAR2);
```

例题 11.4 创建并调用包，完成下列功能：

(1) 在一个包中重载两个过程，分别以学生学号和学生姓名为参数，输出相应学生的基本信息。

(2) 在 PL/SQL 块中调用此包中的过程，实现输出学号为 11432001 的学生的基本信息。

(3) 在 PL/SQL 块中调用此包中的过程，实现输出学生姓名为“张三”的基本信息。

例题解析：

(1) 包的创建。

包说明的创建：

```
CREATE OR REPLACE PACKAGE overload_package
AS
```

```
    PROCEDURE get_student(v_sno number);
    PROCEDURE get_student(v_sname student.sname%TYPE);
END overload_package;
```

包主体的创建：

```
CREATE OR REPLACE PACKAGE BODY overload_package
AS
    PROCEDURE get_student(v_sno number)
    AS
        v_stu student%ROWTYPE;
    BEGIN
        SELECT * INTO v_stu FROM student WHERE sno = v_sno;
        DBMS_OUTPUT.PUT_LINE('学生姓名为:'||v_stu.sname||','||'学生性别:'||v_stu.sex||','||
'学生年龄:'||v_stu.age||'所在系别:'||v_stu.dept);
    END get_student;
    PROCEDURE get_student(v_sname student.sname%TYPE)
    AS
        v_stu student%ROWTYPE;
    BEGIN
        SELECT * INTO v_stu FROM student WHERE sname = v_sname;
        DBMS_OUTPUT.PUT_LINE('学生学号为:'||v_stu.sno ||','||'学生性别:'||v_stu.sex||','||'
学生年龄:'||v_stu.age||'所在系别:'||v_stu.dept);
        END get_student;
END overload_package;
```

（2）包中存储过程的调用。

```
BEGIN
    overload_package.get_student(11432001);
END;
```

程序运行效果如图 11.3 所示。

```
学生姓名为:陈一,学生性别:男  ,学生年龄:17所在系别:侦查系

PL/SQL 过程已成功完成。
```

图 11.3　在包中调用存储过程根据学号查找学生信息

（3）包中存储过程的调用。

```
BEGIN
    overload_package.get_student('张三');
END;
```

程序运行效果如图 11.4 所示。

```
学生学号为:11432003,学生性别:女  ,学生年龄:19所在系别:侦查系
PL/SQL 过程已成功完成。
```

图 11.4 在包中调用存储过程根据姓名查找学生信息

11.4 包的管理

11.4.1 修改包

可以使用 CREATE OR REPLACE PACKAGE 语句重新创建并覆盖原有的包说明，使用 CREATE OR REPLACE PACKAGE BODY 语句重新创建并覆盖原有的包主体。

11.4.2 删除包

可以使用 DROP PACKAGE 语句删除整个包，也可以使用 DROP PACKAGE BODY 语句只删除包主体。当包的说明被删除时，要求包的主体也必须删除，当删除包的主体时，可以不删除包的说明。

例题 11.5 分别删除包 stu_package 的主体部分和说明部分。

例题解析：

```
DROP PACKAGE BODY stu_package;
DROP PACKAGE stu_package;
```

11.4.3 查看语法错误

查看刚编译的包说明或包主体出现错误的详细信息，使用 SHOW ERRORS 命令。

例题 11.6 根据图 11.5 给出的包说明的创建，查看它的语法错误。

```
SQL> CREATE OR REPLACE PACKAGE emp_package
  2  IS
  3  PROCEDURE get_mgr(v_deptno IN emp.deptno%ROWTYPE,
  4                    mgr_ename OUT emp.ename%TYPE);
  5  FUNCTION get_count(v_deptno  emp.deptno%TYPE)
  6  RETURN number;
  7  END;
  8  /
警告: 创建的包带有编译错误。
```

图 11.5 创建包说明的程序运行效果

例题解析：

```
SHOW ERRORS;
```

查看包说明的语法错误，运行效果如图 11.6 所示。

```
PACKAGE EMP_PACKAGE 出现错误:

LINE/COL ERROR
-------- -----------------------------------------------------------
3/1      PL/SQL: Declaration ignored
3/31     PLS-00310: 使用 %ROWTYPE 属性时, 'EMP.DEPTNO' 必须命名表,
         游标或游标变量
```

图 11.6　查看语法错误的程序运行效果

11.4.4　查看结构

通过执行 DESC 命令可以查看包的基本结构，包括包的公有元素、元素的数据类型、包中存储过程的形式参数、形式参数的数据类型、包中存储函数的形式参数、形式参数的数据类型及存储函数的返回值类型等信息。

例题 11.7　查看包 stu_package 的基本结构。

例题解析：

```
DESC stu_package;
```

程序运行效果如图 11.7 所示。

```
FUNCTION GET_GRADE RETURNS NUMBER
参数名称                       类型                    输入/输出默认值?
------------------------------ ----------------------- ------ --------
 V_SNO                          CHAR(8)                 IN
PROCEDURE GET_NUM
参数名称                       类型                    输入/输出默认值?
------------------------------ ----------------------- ------ --------
 V_SNO                          CHAR(8)                 IN
 V_NUM                          NUMBER                  OUT
```

图 11.7　包 stu_package 的基本结构

11.4.5　查看源代码

包的源代码通过查询数据字典 USER_SOURCE 中的 TEXT 即可获得。

```
SELECT TEXT
FROM USER_SOURCE
WHERE NAME = '包名';
```

例题 11.8　查看包 stu_package 的源代码。

例题解析：

```
SELECT TEXT
FROM USER_SOURCE
WHERE NAME = 'STU_PACKAGE';
```

程序运行效果如图 11.8 所示。

```
TEXT
-------------------------------------------------------------
PACKAGE stu_package
IS
PROCEDURE get_num (v_sno IN student.sno%TYPE,v_num OUT NUMBER);
FUNCTION get_grade (v_sno student.sno%TYPE)
RETURN sc.grade%TYPE;
END stu_package;
PACKAGE BODY stu_package
IS
PROCEDURE get_num(v_sno IN student.sno%TYPE,v_num OUT NUMBER)
    IS
    BEGIN
      SELECT COUNT(CNO) INTO v_num FROM sc WHERE sno=v_sno;
END get_num;
  FUNCTION get_grade(v_sno student.sno%TYPE)
  RETURN sc.grade%TYPE
IS
   v_grade sc.grade%TYPE;
  BEGIN
  SELECT AVG(grade) INTO v_grade FROM sc WHERE sno=v_sno;
  RETURN v_grade;
  END get_grade;
END stu_package;

已选择22行。
```

图 11.8 包 stu_package 的源代码

11.5 Oracle 内置包

用户可以开发自己的包,也可以使用 Oracle 提供的包,我们把 Oracle 事先定义的包称为 Oracle 内置包。这些包功能强大,可以供用户使用,每个包针对某一方面的应用。常用的内置包如表 11.2 所示。

表 11.2 常用的 Oracle 内置包

内置包	功 能
DBMS_OUTPUT	从一个存储过程中输出信息
DBMS_MAIL	将 Oracle 系统与 Oracle * Mail 连接起来
DBMS_LOCK	进行复杂的锁机制管理
DBMS_ALERT	标识数据库中发生的某个警告事件
DBMS_PIPE	在不同会话间传递信息(管道通信)
DBMS_JOB	管理作业队列中的作业
DBMS_LOB	操纵大对象(CLOB、BLOB、BFILE 等类型的值)
DBMS_SQL	动态 SQL 语句(通过该包可在 PL/SQL 中执行 DDL 命令)

11.6　实　　验

11.6.1　实验 1　包的创建与调用

1. 实验目的

（1）掌握包说明与包主体的创建方法。

（2）掌握包的调用方法。

2. 实验内容

样本数据库中，员工表 emp 和部门表 dept 的数据信息如图 11.9 和图 11.10 所示。

EMPNO	ENAME	SEX	AGE	JOB	MGR	SALARY	DEPTNO
1001	陈一	男	52	总经理		9500	
2001	姚二	男	47	部门经理	1001	6000	10
2002	张三	男	35	会计	2001	4500	10
2003	李四	女	27	出纳	2001	3500	10
3001	王五	女	38	部门经理	1001	5500	20
3002	赵六	女	27	文员	3001	2500	20
3003	陈七	女	23	文员	3001	1800	20
4001	刘八	男	40	部门经理	1001	6500	30
4002	张九	男	35	业务员	4001	3500	30
4003	孙十	女	24	业务员	4001	2400	30
5001	沈一	男	38	部门经理	1001	6500	40
5002	吴二	女	32	程序员	5001	3500	40
5003	崔三	男	27	程序员	5001	3000	40
5004	刘四	男	19	程序员	5001	1800	40
6001	张五	女	35	部门经理	1001	4000	50
6002	韩六	男	26	维修员	6001	2200	50

图 11.9　员工表 emp 中的数据

DEPTNO	DNAME	LOC
10	财务部	上海
20	人力资源部	北京
30	销售部	北京
40	研发部	上海
50	客服部	大连

图 11.10　部门表 dept 中的数据

创建并调用包，完成下列功能：

（1）创建一个包，包名为 emp_package。其中包括一个存储过程，根据部门编号返回该部门的经理；还包括一个存储函数，根据部门编号返回该部门的员工人数。

（2）在 PL/SQL 块中调用此包中的过程，实现输出部门编号为 30 的部门经理的姓名。

（3）在 PL/SQL 块中调用此包中的函数，实现输出部门编号为 30 的员工人数。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，创建包说明和包主体

部分,并按照题目要求进行调用实现相应程序功能。参数定义准确,程序无语法错误,书写规范,运行无误为优秀。如果出现错误,则根据错误点数以及结构合理性灵活给分。

11.6.2 实验2 包的重载

1. 实验目的

(1) 掌握包说明与包主体的创建方法。

(2) 掌握包的调用方法。

(3) 掌握包的重载方法及注意事项。

2. 实验内容

样本数据库中,员工表 emp 和部门表 dept 的数据信息如图 11.11 和图 11.12 所示。

EMPNO	ENAME	SEX	AGE	JOB	MGR	SALARY	DEPTNO
1001	陈一	男	52	总经理		9500	
2001	姚二	男	47	部门经理	1001	6000	10
2002	张三	男	35	会计	2001	4500	10
2003	李四	女	27	出纳	2001	3500	10
3001	王五	女	38	部门经理	1001	5500	20
3002	赵六	女	27	文员	3001	2500	20
3003	陈七	女	23	文员	3001	1800	20
4001	刘八	男	40	部门经理	1001	6500	30
4002	张九	男	35	业务员	4001	3500	30
4003	孙十	女	24	业务员	4001	2400	30
5001	沈一	男	38	部门经理	1001	6500	40
5002	吴二	女	32	程序员	5001	3500	40
5003	崔三	男	27	程序员	5001	3000	40
5004	刘四	男	19	程序员	5001	1800	40
6001	张五	女	35	部门经理	1001	4000	50
6002	韩六	男	26	维修员	6001	2200	50

图 11.11 员工表 emp 中的数据

DEPTNO	DNAME	LOC
10	财务部	上海
20	人力资源部	北京
30	销售部	北京
40	研发部	上海
50	客服部	大连

图 11.12 部门表 dept 中的数据

创建并调用包,完成下列功能:

(1) 在一个包中重载两个过程,分别以部门号和部门名称为参数,查询相应部门的信息。

(2) 在 PL/SQL 块中调用此包中的过程,实现输出部门号为 30 的部门信息。

(3) 在 PL/SQL 块中调用此包中的过程,实现输出部门名称为"财务部"的部门信息。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,创建包说明和包主体

部分,并按照题目要求进行调用实现相应程序功能。参数定义准确,程序无语法错误,书写规范,运行无误为优秀。如果出现错误,则根据错误点数以及结构合理性灵活给分。

11.7　本章小结

本章首先介绍了包的定义和包的优点,与存储过程和函数一样,包被存放在数据库的数据字典中。当完成一项复杂的工作任务时,可以把多个存储过程或者函数集成到一个包中,以便于对存储过程和函数的管理。

其次介绍了创建包的步骤、语法格式及包的调用方法。程序包的创建主要包括两部分,分别是包说明部分的创建和包主体的创建。在包的调用过程中主要区分如何使用公有元素、私有元素和局部元素,各种元素的使用规则,以便更好地使程序模块化,并且做到信息隐藏。

最后介绍了包的重载,包括重载的形式、重载的规则及重载的方法,包的各种管理命令和常用的 Oracle 内置包的功能。

11.8　课后习题

一、选择题

1. 以下有关包的说法不正确的是(　　)。
   A. 创建包分为两个部分,分别为包说明和包主体
   B. 包主体是包说明部分的具体实现,它们的创建不分先后
   C. 包说明部分声明的元素为公有元素,可以被包外的其他 PL/SQL 程序块访问
   D. 包主体部分声明的元素为私有元素,只有同一个包中的过程和函数才能访问
2. 以下有关包的说法正确的是(　　)。
   A. 在 Oracle 数据库中,包有两类:一类是系统包,另一类是用户创建的包
   B. 如果包声明中不包含任何过程和函数,也必须创建包主体
   C. 包的创建分为两个步骤:分别是包说明的创建和包主体的创建,这两部分作为一个整体存放在数据库的字典中
   D. 包中的过程或函数被调用时,必须在过程或函数名前面加上包的名字
3. 关于包的管理命令,说法不正确的是(　　)。
   A. 可以使用 DROP PACKAGE 语句删除整个包
   B. 可以使用 DROP PACKAGE BODY 语句只删除包主体
   C. 当包说明被删除时,可以不删除包主体
   D. 当包主体被删除时,可以不删除包说明

二、应用题

1. 创建并调用包,完成下列功能:

(1) 定义一个程序包,该程序包包括一个函数和一个过程。函数以部门号为参数返回该部门的平均工资;过程以部门号为参数,输出该部门中工资低于平均工资的员工的员工号、员工名。

(2) 在 PL/SQL 块中调用包中的过程和函数,测试并输出编号为 30 的部门平均工资和工资低于平均工资的员工信息。

2. 创建并调用包,完成下列功能:

(1) 在一个包中重载两个过程,分别以课程号和学分为参数,输出相应课程的基本信息。

(2) 在 PL/SQL 块中调用此包中的过程,实现输出课程号为 c4 的课程基本信息。

(3) 在 PL/SQL 块中调用此包中的过程,实现输出学分为 3 的课程基本信息。

第 12 章

触 发 器

12.1 触发器概述

12.1.1 触发器的概念

触发器类似于存储过程和存储函数，都是有声明、执行和异常处理过程的 PL/SQL 有名块，触发器编译后存储在数据库的数据字典中，但是用户不能直接调用触发器。当一个表或者视图被改变时，或者数据库中发生某些事件时，Oracle 会自动激发触发器，并执行触发器中的代码。触发器不允许用户显示传递参数，不能够返回参数值，也不允许用户调用触发器，触发器只能由 Oracle 在合适的时机自动调用。

运行触发器的方式叫作激发触发器，能够激发触发器的事件有：DML 操作(INSERT、UPDATE、DELETE)、DDL 操作、系统事件(如数据库的关闭和启动)、用户事件(如用户登录)等。

激发触发器的语句叫作触发语句。如果我们执行的一条 SQL 语句激发了一个触发器，我们就把这条 SQL 语句叫作触发语句。

12.1.2 触发器的作用

触发器具有更精细和更复杂的数据控制能力，主要作用如下。

1. 提高安全性

触发器可以根据数据库的值使用户具有操作数据库的某种权利。比如，触发器可以基于时间或者数据库汇总的数据限制用户的操作。

2. 具有审计功能

触发器可以跟踪用户对数据库的操作。比如，审计用户操作数据库的语句，把用户对数据库的更新写入审计表。

3. 实现复杂的数据库完整性规则

触发器可以实现非标准的数据完整性检查约束，可产生比规则更复杂的限制，并可以提供可变的默认值。

触发器可以实现非标准的数据库相关完整性规则，可以对数据库中相关的表进行连环更新，并且能够拒绝或者回退哪些破坏相关完整性的变化，取消试图进行的数据更新的事务。

4. 同步实时的复制表中的数据

无论数据库中表的数据如何变化，都可以通过触发器实时的将表中数据复制出来。

5. 自动计算数据值

如果数据的值达到了一定的要求，可以通过触发器进行特定的处理。

12.1.3 触发器的类型

触发器按照触发事件类型和对象不同，可以分为以下几类：语句级触发器、行级触发器、INSTEAD OF 触发器、系统事件触发器和用户事件触发器。

1. 语句级触发器

语句级触发器(Statement Trigger)是基于语句级的，当一条 SQL 语句改变数据时，无论这条 SQL 语句影响了多少条记录，语句级触发器都只触发一次。SQL 语句执行一次，语句级触发器就被触发一次。

2. 行级触发器

行级触发器(Row Trigger)的触发机制是基于行的。当表中数据改变时，将激发行级触发器，改变一行数据，就触发一次。如果改变 n 行数据，则触发器将被触发 n 次。

3. INSTEAD OF 触发器

在某些视图上，不能直接对其进行更新操作，但可以在这种视图上建立触发器，利用触发器对视图的基本表进行更新操作，这类触发器叫作 INSTEAD OF 触发器。

4. 系统事件触发器

系统事件触发器与表、视图没有关系。系统事件包括数据库启动、关闭、服务器错误、数据库角色改变等。当这些事件发生时，就会激发系统事件触发器。可以通过系统事件触发器实现对数据库的审计功能。

5. 用户事件触发器

用户事件包括用户登录数据库、用户退出数据库、用户执行 DDL 语句和 DML 语句等。当这些事件发生时，就会激发用户事件触发器。

12.1.4 触发器的组成

触发器创建之前，必须先确定好其触发时间、触发事件以及触发器的类型。触发器的组成如表 12.1 所示。

表 12.1　触发器的组成

组成部分	描　　述	可　能　值
作用对象	触发器作用的对象	表,数据库,视图,模式
触发事件	触动触发器的数据操作类型	DML,DDL,数据库系统事件
触发时间	与触发事件的时间次序	BEFORE,AFTER
触发级别	触发器体被执行的次数	STATEMENT,ROW
触发条件	选择性执行触发事件的条件	TRUE,FALSE
触发器体	该触发器将要执行的动作	完整的 PL/SQL 块

(1) 作用对象：触发器作用的对象包括表、视图、数据库和模式。

(2) 触发事件：即在何种情况下激发触发器。如 DML（INSERT、UPDATE、DELETE）、DDL、数据库系统事件等,可以将多个事件用关系运算符 OR 组合。

(3) 触发时间：BEFORE 和 AFTER 指出触发器的触发时序分别为前触发和后触发方式,前触发是在执行触发事件之前触发当前所创建的触发器,后触发是在执行触发事件之后触发当前所创建的触发器。

(4) 触发级别：触发器默认为语句级触发器。语句级触发器将整个语句操作作为触发事件,当它符合约束条件时,激活一次触发器,触发器只执行一次。

用 FOR EACH ROW 选项说明的触发器为行级触发器。行级触发器要求当一个 DML 语句操作影响表中的多行数据时,对于其中的每一行数据,只要它们符合触发条件,均激活一次触发器,触发器执行多次。

(5) 触发条件：由 WHEN 子句指定一个逻辑表达式,当触发事件发生,而且 WHEN 条件为 TRUE 时,触发器才会执行。

(6) 触发器体：触发器执行时所进行的操作。

12.2　语句级触发器

12.2.1　语句级触发器的创建

通过 CREATE TRIGGER 语句创建一个语句级触发器,该触发器在一个数据操作语句 DML 发生时只触发一次。

语句级触发器创建的语法如下：

```
CREATE [OR REPLACE] TRIGGER trigger_name
[BEFORE|AFTER] trigger_event1 [OR trigger_event2 … ] [OF column_name]
ON table_name
PL/SQL block
```

说明：

(1) trigger_name 指触发器名，触发器存在于单独的名字空间中，可以与其他对象同名，而过程、函数、包具有相同的名字空间，因此相互间不能同名。

(2) trigger_event 指明触发事件的数据操纵语句，取值为 INSERT 或 UPDATE 或 DELETE。

(3) column_name 指明表中的某一个属性列，用 UPDATE OF column_name 指明 UPDATE 事件只有在修改特定列时才触发，否则修改任何一列都触发。

(4) table_name 指明与该触发器相关联的表名。

(5) PL/SQL block 指触发器体，指明该触发器将执行的操作。

例题 12.1 创建一个语句级触发器，当执行删除员工表 emp 中员工信息操作后，输出提示信息"您执行了删除操作…"。

例题解析：

```
CREATE OR REPLACE TRIGGER delete_trigger1
AFTER DELETE ON emp
BEGIN
    DBMS_OUTPUT.PUT_LINE('您执行了删除操作…');
END delete_trigger1;
```

测试触发器语句：

```
DELETE FROM emp WHERE deptno = 30;
```

测试该触发器的程序运行效果如图 12.1 所示。

```
SQL> DELETE FROM emp WHERE deptno=30;
您执行了删除操作…

已删除3行。
```

图 12.1 测试语句级触发器

例题 12.2 创建一个语句级触发器，当执行更新学生表 student 中学生年龄操作时，统计更新后所有学生的平均年龄并输出。

例题解析：

```
CREATE OR REPLACE TRIGGER update_trigger2
AFTER UPDATE OF age ON student
DECLARE
    v_avg NUMBER;
BEGIN
    SELECT AVG(age) INTO v_avg FROM student;
    DBMS_OUTPUT.PUT_LINE('更新后学生的平均年龄为:'||v_avg);
END update_trigger2;
```

测试触发器语句：

```
UPDATE student SET age = age + 1 WHERE dept = '侦查系';
```

测试该触发器的程序运行效果如图 12.2 所示。

```
SQL> UPDATE student SET age=age+1 WHERE dept='侦查系';
更新后学生的平均年龄为:20.5

已更新3行。
```

图 12.2　测试语句级触发器

12.2.2　触发器谓词

如果触发器响应多个 DML 事件，而且需要根据事件的不同进行不同的操作，则可以在触发器体中使用谓词判断是哪个触发事件触动了触发器。触发器谓词的行为和值如表 12.2 所示。

表 12.2　触发器谓词的行为和值

谓　　词	行　　为
INSERTING	如果触发事件是 INSERT 操作，则谓词的值为 TRUE，否则为 FALSE
UPDATING	如果触发事件是 UPDATE 操作，则谓词的值为 TRUE，否则为 FALSE
DELETING	如果触发事件是 DELETE 操作，则谓词的值为 TRUE，否则为 FALSE

例题 12.3　创建一个 AFTER 型语句级触发器，当对员工表 emp 执行插入操作时，统计插入操作后员工人数并输出；当对员工表 emp 执行更新操作时，统计更新操作后员工平均工资并输出；当对员工表 emp 执行删除操作时，统计删除后员工最小年龄并输出。

例题解析：

```
CREATE OR REPLACE TRIGGER emp_trigger3
AFTER INSERT OR UPDATE OR DELETE ON emp
DECLARE
    v_count NUMBER;
    v_sal emp.salary%TYPE;
    v_age emp.age%TYPE;
BEGIN
    IF INSERTING THEN
        SELECT COUNT( * ) INTO v_count FROM emp;
        DBMS_OUTPUT.PUT_LINE('员工人数为:'||v_count);
    END IF;
    IF UPDATING THEN
        SELECT AVG(salary) INTO v_sal FROM emp;
        DBMS_OUTPUT.PUT_LINE('员工平均工资为:'||v_sal);
```

```
    END IF;
    IF DELETING THEN
        SELECT MIN(age) INTO v_age FROM emp;
        DBMS_OUTPUT.PUT_LINE('员工最小年龄为:'||v_age);
    END IF;
END emp_trigger3;
```

测试触发器语句：

```
INSERT INTO emp VALUES('11432013','杨珺婷','女',25,'会计主管','1001',6500,'10');
```

测试该触发器的程序运行效果如图 12.3 所示。

```
SQL> INSERT INTO emp VALUES('11432013','杨珺婷','女',25,'会计主管','1001',6500,'10');
员工人数为:17

已创建 1 行。
```

图 12.3　利用数据插入测试语句级触发器

```
UPDATE emp SET salary = salary + 800 WHERE deptno = 20;
```

测试该触发器的程序运行效果如图 12.4 所示。

```
SQL> UPDATE emp SET salary=salary+800 WHERE deptno=20;
员工平均工资为:4318.75

已更新3行。
```

图 12.4　利用数据更新测试语句级触发器

```
DELETE FROM emp WHERE deptno = 40;
```

测试该触发器的程序运行效果如图 12.5 所示。

```
SQL> DELETE FROM emp WHERE deptno=40;
员工最小年龄为:23

已删除4行。
```

图 12.5　利用数据删除测试语句级触发器

12.3　行级触发器

12.3.1　行级触发器的创建

在创建触发器时，如果使用了 FOR EACH ROW 选项，则表示该触发器为行级触发器。行级触发器和语句级触发器的区别表现在：当一个 DML 语句操作影响数据库中的多行数据时，对于其中的每个数据行，行级触发器均会被触发一次。

行级触发器创建的语法如下：

```
CREATE [OR REPLACE] TRIGGER trigger_name
[BEFORE|AFTER] trigger_event1 [OR trigger_event2…] [OF column_name]
ON table_name
FOR EACH ROW
[WHEN trigger_condition]
PL/SQL block
```

说明：trigger_condition 为指定的限制条件，以确定触发器体是否被执行。在触发事件发生并满足此限制条件时，触发器体执行；否则，触发器体不被执行。

例题 12.4　创建一个行级触发器，当更新学生表 student 中学生信息后，输出提示信息“您执行了更新操作…”。

例题解析：

```
CREATE OR REPLACE TRIGGER update_trigger4
AFTER UPDATE ON student
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('您执行了更新操作…');
END update_trigger4;
```

测试触发器语句：

```
UPDATE student SET age = age - 1 WHERE sex = '男';
```

测试该触发器的程序运行效果如图 12.6 所示。

```
SQL> UPDATE student SET age=age-1 WHERE sex='男';
您执行了更新操作…
您执行了更新操作…
您执行了更新操作…
您执行了更新操作…
您执行了更新操作…

已更新5行。
```

图 12.6　测试行级触发器

12.3.2　触发器标识符

当编写触发器时，如果需要引用被插入和被删除记录的值，或者被更新记录更新前和更新后的值，标识符:old 和:new 就是为这种用途提供的。

在行级触发器中，在列名前加上:old 标识符表示该列变化前的值，在列名前加上:new 标识符表示该列变化后的值。:old 和:new 标识符的含义如表 12.3 所示。

表 12.3　:old 和 :new 标识符的含义

触发事件	:old. 列名	:new. 列名
INSERT	所有字段都是 NULL	当该语句完成时将要插入的数值
UPDATE	在更新之前该列的原始值	当该语句完成时将要更新的新值
DELETE	在删除行之前该列的原始值	所有字段都是 NULL

说明：

(1) 在行级触发器中使用这些标识符。

(2) 在语句级触发器中不要使用这些标识符。

(3) 在触发器体的 SQL 语句或 PL/SQL 语句中使用这些标识符时，前面要加“:”。

(4) 在行级触发器的 WHEN 限制条件中使用这些标识符时，前面不要加“:”。

例题 12.5　创建一个行级 UPDATE 触发器，当更新学生表 student 中某个学生的系别名称时，激发触发器，输出该学生的学号以及修改前的系别名称与修改后的系别名称。

例题解析：

```
CREATE OR REPLACE TRIGGER update_trigger5
AFTER UPDATE OF dept ON student
FOR EACH ROW
BEGIN
DBMS_OUTPUT.PUT_LINE('学生的学号为:'||:old.sno);
DBMS_OUTPUT.PUT_LINE('修改前的系别名称为:'||:old.dept);
DBMS_OUTPUT.PUT_LINE('修改后的系别名称为:'||:new.dept);
END update_trigger5;
```

测试触发器语句：

```
UPDATE student SET dept = '监所管理系' WHERE age > 20;
```

测试该触发器的程序运行效果如图 12.7 所示。

例题 12.6　创建一个行级 DELETE 触发器，当删除课程表 course 中某门课程信息时，激发触发器，同时删除选课表 sc 中该课程的所有选修记录。

例题解析：

```
CREATE OR REPLACE TRIGGER delete_trigger6
BEFORE DELETE ON course
FOR EACH ROW
BEGIN
    DELETE FROM sc WHERE cno = :old.cno;
END delete_trigger6;
```

测试触发器语句：

```
DELETE FROM course WHERE cno = 'c2';
```

测试该触发器的程序运行效果如图 12.8 所示。

```
SQL> UPDATE student SET dept='监所管理系' WHERE age>20;
学生的学号为:11432002
修改前的系别名称为:侦查系
修改后的系别名称为:监所管理系
学生的学号为:11432004
修改前的系别名称为:刑事技术系
修改后的系别名称为:监所管理系
学生的学号为:11432005
修改前的系别名称为:刑事技术系
修改后的系别名称为:监所管理系
学生的学号为:11432007
修改前的系别名称为:公安信息系
修改后的系别名称为:监所管理系
学生的学号为:11432008
修改前的系别名称为:公安信息系
修改后的系别名称为:监所管理系
学生的学号为:11432010
修改前的系别名称为:治安管理系
修改后的系别名称为:监所管理系

已更新6行。
```

图 12.7　利用标识符测试行级触发器

```
SQL> SELECT COUNT(*) FROM sc WHERE cno='c2';

  COUNT(*)
----------
         9

SQL> DELETE FROM course WHERE cno='c2';

已删除 1 行。

SQL> SELECT COUNT(*) FROM sc WHERE cno='c2';

  COUNT(*)
----------
         0
```

图 12.8　利用标识符测试行级触发器

12.3.3　触发器的 WHEN 子句

在行级触发器中使用 WHEN 子句,可以进一步控制触发器的执行。保证当行级触发器被触发时只有在当前行满足一定限制条件时,才执行触发器体的 PL/SQL 语句。

WHEN 子句后面是一个逻辑表达式,当逻辑表达式的值为 TRUE 时,执行触发器体。如果逻辑表达式为 FALSE,不执行触发器体。

例题 12.7　使用 WHEN 子句创建一个行级触发器,修改员工工资时,保证修改后的工资

高于修改前的工资，否则请利用 RAISE_APPLICATION_ERROR 过程抛出错误号为“－20001”的错误，错误信息为“修改后的工资低于修改前的工资，钱不够花呀！”。

例题解析：

```
CREATE OR REPLACE TRIGGER update_trigger7
BEFORE UPDATE OF salary ON emp
FOR EACH ROW
WHEN(new.salary<=old.salary)
BEGIN
    RAISE_APPLICATION_ERROR(-20001, '修改后的工资低于修改前的工资,钱不够花呀!');
END update_trigger7;
```

测试触发器语句：

```
UPDATE emp SET salary=1500 WHERE empno='3002';
```

测试该触发器的程序运行效果如图 12.9 所示。

```
SQL> UPDATE emp SET salary=1500 WHERE empno='3002';
UPDATE emp SET salary=1500 WHERE empno='3002'
       *
第 1 行出现错误:
ORA-20001: 修改后的工资低于修改前的工资，钱不够花呀!
ORA-06512: 在 "SYSTEM.UPDATE_TRIGGER7", line 2
ORA-04088: 触发器 'SYSTEM.UPDATE_TRIGGER7' 执行过程中出错
```

图 12.9　利用 WHEN 子句测试行级触发器

说明：在触发器体中，通过 RAISE_APPLICATION_ERROR 存储过程抛出一个错误，给出错误号以及错误提示信息。通过 WHEN 子句，使用行级触发器标识符表示修改前和修改后的工资，并进行大小比较，如果逻辑表达式为真，则执行触发器体。

12.4　INSTEAD OF 触发器

12.4.1　INSTEAD OF 触发器的作用

INSTEAD OF 触发器的主要作用是修改一个本来不可以被修改的视图。INSTEAD OF 触发器是建立在视图上的触发器，响应视图上的 DML 操作。由于对视图的 DML 操作最终会转换为对基本表的操作，因此激发 INSTEAD OF 触发器的 DML 语句本身并不执行，而是转换到触发器体中处理，所以这种类型的触发器被称为 INSTEAD OF（替代）触发器。此外，INSTEAD OF 触发器必须是行级触发器。

12.4.2　触发器的创建

INSTEAD OF 触发器创建的语法如下：

```
CREATE [OR REPLACE] TRIGGER trigger_name
INSTEAD OF trigger_event1 [OR trigger_event2 … ] [OF column_name]
ON view_name
FOR EACH ROW
[WHEN trigger_condition]
PL/SQL block
```

说明：

(1) INSTEAD OF 是关键字，替换其他类型触发器中的 BEFORE 或 AFTER 标识符。

(2) view_name 是视图的名字，替换其他类型触发器中的 table_name(表名)。

例题 12.8　INSTEAD OF 触发器的应用。

首先已创建了一个基于学生表 student 和选课表 sc 两个表连接的视图 stu_sc，如图 12.10 所示。

```
SQL> CREATE VIEW stu_sc
  2  AS SELECT student.sno,sname,sex,age,dept,cno,grade
  3     FROM student,sc
  4     WHERE student.sno=sc.sno;

视图已创建。
```

图 12.10　视图 stu_sc 的创建

向视图 stu_sc 中插入一条记录，运行结果如图 12.11 所示。

```
SQL> INSERT INTO stu_sc
  2  VALUES('11432015','小新','男',19,'侦查系','c1',90);
INSERT INTO stu_sc
*
第 1 行出现错误:
ORA-01779: 无法修改与非键值保存表对应的列
```

图 12.11　创建 INSTEAD OF 触发器之前向视图 stu_sc 中插入数据

当视图是在多表连接的基础上创建的，无法对视图进行 DML 操作，因为对视图的 DML 操作就是对基本表进行 DML 操作，而此时创建视图的基本表是多个，所以该操作并不能确定是在哪个基本表上进行的，所以会产生错误。在视图 stu_sc 上创建一个 INSTEAD OF 触发器，解决上面不能对视图进行 DML 操作的问题。

例题解析：

```
CREATE OR REPLACE TRIGGER view_trigger8
INSTEAD OF INSERT ON stu_sc
FOR EACH ROW
BEGIN
    INSERT INTO student(sno,sname,sex,age,dept)
    VALUES(:new.sno,:new.sname,:new.sex,:new.age,:new.dept);
    INSERT INTO sc(sno,cno,grade)
    VALUES(:new.sno,:new.cno,:new.grade);
```

```
END view_trigger8;
```

在触发器创建之后，再次执行上面的插入操作，插入语句的运行结果如图 12.12 所示。学生表和选课表中新增的数据如图 12.13 和图 12.14 所示。

```
SQL> INSERT INTO stu_sc
  2  VALUES('11432015','小新','男',19,'侦查系','c1',90);

已创建 1 行。
```

图 12.12 创建 INSTEAD OF 触发器之后向视图 stu_sc 中插入数据

```
SQL> SELECT * FROM student WHERE sno='11432015';

SNO      SNAME                SEX        AGE DEPT
-------- -------------------- ---- ---------- ------
11432015 小新                 男           19 侦查系
```

图 12.13 学生表中新增的数据

```
SQL> SELECT * FROM sc WHERE sno='11432015';

SNO      CNO           GRADE
-------- -------- ----------
11432015 c1               90
```

图 12.14 选课表中新增的数据

12.5 系统事件与用户事件触发器

12.5.1 系统事件与用户事件

系统事件是指 Oracle 数据库本身的动作所触发的事件。这些事件主要包括数据库启动、数据库关闭、系统错误等。

用户事件是相对于用户的所执行的表(视图)等 DML 操作而言的。常见的用户事件包括 CREATE 事件、TRUNCATE 事件、DROP 事件、ALTER 事件、COMMIT 事件和 ROLLBACK 事件。系统事件与用户事件触发器不是常用的触发器。

12.5.2 触发器的创建

触发器创建的语法如下：

```
CREATE [OR REPLACE] TRIGGER trigger_name
[BEFORE|AFTER] trigger_event1 [OR trigger_event2 … ] [OF column_name]
ON [DATABASE|SCHEMA]
[WHEN trigger_condition]
PL/SQL block
```

说明：

(1) 系统事件触发器的作用对象是数据库 DATABASE。

(2) 用户事件触发器的作用对象一般是 USER. SCHEMA，即将触发器建立在该用户及用户所拥有的所有对象之上。

例题 12.9　创建一个系统事件触发器，当数据库启动时，自动地将启动时间记录到日志表 database_log 中。创建一个日志表 database_log 用来存放数据库的启动时间，如图 12.15 所示。

```
SQL> CREATE TABLE database_log
  2  (startup_date TIMESTAMP);

表已创建。
```

图 12.15　日志表 database_log 的创建

例题解析：

```
CREATE OR REPLACE TRIGGER startup_trigger9
AFTER STARTUP ON DATABASE
BEGIN
    INSERT INTO database_log VALUES(sysdate);
END startup_trigger9;
```

启动数据库后，查询日志表 database_log 中的信息如图 12.16 所示。

```
SQL> SELECT * FROM database_log;

STARTUP_DATE
-----------------------------------
19-10月-14 03.23.16.000000 下午
```

图 12.16　日志表 database_log 中的信息

例题 12.10　创建一个用户事件触发器，它会在任何人登录到创建该触发器的模式时触发，自动地将用户名和登录时间记录到日志表 login_table 中。创建一个日志表 login_table 用来存放用户名和登录时间，如图 12.17 所示。

```
SQL> CREATE TABLE login_table
  2  (username VARCHAR2(20),
  3  log_date TIMESTAMP);

表已创建。
```

图 12.17　日志表 login_table 的创建

例题解析：

```
CREATE OR REPLACE TRIGGER log_trigger10
AFTER LOGON ON SCHEMA
```

```
BEGIN
    INSERT INTO login_table VALUES(user,sysdate);
END log_trigger10;
```

查看日志表 login_table 中的信息如图 12.18 所示。

```
SQL> SELECT * FROM login_table;

USERNAME             LOG_DATE
-------------------- ------------------------------
SYSTEM               19-10月-14 03.30.44.000000 下午
```

图 12.18　日志表 login_table 中的信息

12.6　触发器的管理

12.6.1　修改触发器

可以使用 CREATE OR REPLACE TRIGGER 语句重新创建并覆盖原有的触发器。

12.6.2　禁用触发器

可以使用 ALTER TRIGGER 触发器名 DISABLE 语句禁用某个触发器。

可以使用 ALTER TRIGGER 表名 DISABLE ALL TRIGGER 禁用某个表对象上的所有触发器。

例如，禁用触发器 delete_trigger1。

```
ALTER TRIGGER delete_trigger1 DISABLE;
```

12.6.3　启用触发器

可以使用 ALTER TRIGGER 触发器名 ENABLE 语句启用某个触发器。

可以使用 ALTER TRIGGER 表名 ENABLE ALL TRIGGER 启用某个表对象上的所有触发器。

例如，启用触发器 delete_trigger1。

```
ALTER TRIGGER delete_trigger1 ENABLE;
```

12.6.4　删除触发器

可以使用 DROP TRIGGER 语句删除触发器。

例如，删除触发器 delete_trigger1。

```
DROP TRIGGER delete_trigger1;
```

12.6.5　查看语法错误

查看刚编译的触发器出现错误的详细信息，可以使用 SHOW ERRORS 命令。

例题 12.11　根据图 12.19 给出的触发器的定义，查看它的语法错误。

```
SQL> CREATE OR REPLACE TRIGGER update_trigger11
  2  AFTER UPDATE OF dept ON student
  3  FOR EACH ROW
  4  BEGIN
  5  DBMS_OUTPUT.PUT_LINE('学生的学号为:'||old.sno);
  6  DBMS_OUTPUT.PUT_LINE('修改前的系别名称为:'||old.dept);
  7  DBMS_OUTPUT.PUT_LINE('修改后的系别名称为:'||new.dept);
  8  END update_trigger11;
  9  /

警告: 创建的触发器带有编译错误。
```

图 12.19　创建触发器的程序运行效果

例题解析：

```
SHOW ERRORS;
```

查看触发器的语法错误，运行效果如图 12.20 所示。

```
SQL> show errors;
TRIGGER UPDATE_TRIGGER11 出现错误:

LINE/COL ERROR
-------- -----------------------------------
2/1      PL/SQL: Statement ignored
2/39     PLS-00201: 必须声明标识符 'OLD.SNO'
3/1      PL/SQL: Statement ignored
3/45     PLS-00201: 必须声明标识符 'OLD.DEPT'
4/1      PL/SQL: Statement ignored
4/48     PLS-00201: 必须声明标识符 'NEW.DEPT'
```

图 12.20　查看语法错误的程序运行效果

12.6.6　查看源代码

触发器的源代码通过查询数据字典 USER_SOURCE 中的 TEXT 即可获得。

```
SELECT TEXT
FROM USER_SOURCE
WHERE NAME = '触发器名';
```

例题 12.12　查看触发器 delete_trigger6 的源代码。

例题解析：

```
SELECT TEXT
FROM USER_SOURCE
WHERE NAME = 'DELETE_TRIGGER6'
```

程序运行效果如图 12.21 所示。

```
TEXT
------------------------------------------

TRIGGER delete_trigger6
BEFORE DELETE ON dept
FOR EACH ROW
BEGIN
DELETE FROM emp WHERE deptno=:old.deptno;
END delete_trigger6;

已选择6行。
```

图 12.21 触发器 DELETE_TRIGGER6 的源代码

12.7 实 验

12.7.1 实验 1 语句级触发器

1. 实验目的

(1) 了解触发器的组成。

(2) 掌握语句级触发器的创建与测试方法。

2. 实验内容

样本数据库中,学生表 student 的数据信息如图 12.22 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 12.22 学生表 student 中的数据

(1) 创建一个语句级触发器,当在学生表 student 中更新学生信息后,输出提示信息“您执行了更新数据的操作…”。

(2) 为学生表 student 创建一个触发器。当执行插入操作时,输出插入后的学生总人

数；当执行更新年龄操作时，输出更新后的学生平均年龄；当执行删除操作时，输出删除后的学生最大年龄。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据用户需求创建满足条件的触发器，并且对触发器进行测试其结果应该是正确的，程序无语法错误，书写规范，运行无误为优秀。如果出现错误，则根据错误点数以及结构合理性灵活给分。

12.7.2　实验 2　行级触发器

1. 实验目的

(1) 了解触发器的组成。

(2) 掌握行级触发器的创建与测试方法。

2. 实验内容

样本数据库中，员工表 emp 和部门表 dept 的数据信息如图 12.23 和图 12.24 所示。

EMPNO	ENAME	SEX	AGE	JOB	MGR	SALARY	DEPTNO
1001	陈一	男	52	总经理		9500	
2001	姚二	男	47	部门经理	1001	6000	10
2002	张三	男	35	会计	2001	4500	10
2003	李四	女	27	出纳	2001	3500	10
3001	王五	女	38	部门经理	1001	5500	20
3002	赵六	女	27	文员	3001	2500	20
3003	陈七	女	23	文员	3001	1800	20
4001	刘八	男	40	部门经理	1001	6500	30
4002	张九	男	35	业务员	4001	3500	30
4003	孙十	女	24	业务员	4001	2400	30
5001	沈一	男	38	部门经理	1001	6500	40
5002	吴二	女	32	程序员	5001	3500	40
5003	崔三	男	27	程序员	5001	3000	40
5004	刘四	男	19	程序员	5001	1800	40
6001	张五	女	35	部门经理	1001	4000	50
6002	韩六	男	26	维修员	6001	2200	50

图 12.23　员工表 emp 中的数据

DEPTNO	DNAME	LOC
10	财务部	上海
20	人力资源部	北京
30	销售部	北京
40	研发部	上海
50	客服部	大连

图 12.24　部门表 dept 中的数据

(1) 创建一个行级 UPDATE 触发器，当更新员工表 emp 中某个员工的工资时，激发触发器，输出该员工的姓名以及修改前的工资与修改后的工资。

(2) 创建一个行级 DELETE 触发器，当删除部门表 dept 中某个部门信息时，激发触发器，同时删除员工表 emp 中该部门员工的详细信息。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据用户需求创建满足条件的触发器，并且对触发器进行测试其结果应该是正确的，程序无语法错误，书写规范，运行无误为优秀。如果出现错误，则根据错误点数以及结构合理性灵活给分。

12.8 本章小结

本章首先介绍了触发器的概念与作用、触发器的类型、触发器的组成，包括作用对象、触发事件、触发时间、触发级别、触发条件和触发体等。

其次重点介绍了语句级触发器和行级触发器的开发步骤、创建语法、触发事件的谓词、触发器标识符、触发条件 WHEN 子句的应用等。

最后介绍了 INSTEAD OF 触发器、系统事件与用户事件触发器的功能、创建语法，以及触发器的各种管理命令，包括如何修改触发器、禁用触发器、启用触发器、删除触发器、查看语法错误和查看源代码。

12.9 课后习题

一、选择题

1. 下列哪一个动作不会激发一个 DML 触发器？(　　)

   A. 更新数据　　B. 查询数据　　C. 删除数据　　D. 插入数据

2. 以下关于触发器中的标识符说法，正确的是(　　)。

   A. :old 和:new 是语句级触发器的标识符

   B. 对于 UPDATE 触发事件，:old 和:new 有效

   C. 对于 INSERT 触发事件，:old 有效

   D. 对于 DELETE 触发事件，:new 有效

3. 以下关于触发器说法，不正确的是(　　)。

   A. 触发器不可以接受参数

   B. 触发器内禁止使用 COMMIT 或 ROLLBACK 语句

   C. 触发器中不能对数据进行增、删、改、查

   D. 触发器不能被调用，只能被触发

二、应用题

1. 创建一个语句级触发器,当向学生表执行插入操作时,统计插入后的学生平均年龄并输出;当执行删除操作时,统计删除后的女学生的总人数并输出。

2. 创建一个行级 UPDATE 触发器,当更新学生表 student 中的学号时,激发触发器,自动修改选课信息表 SC 中学生的学号。

3. 编写一个触发器,当修改 emp 表中的工资时,保证编号为 30 的部门不超过 8000 元,其他部门不超过 10 000 元。

第 13 章

数据库安全性与完整性

13.1　数据库安全性概述

13.1.1　安全控制模型

数据库在各种信息系统中得到广泛的应用，数据在信息系统中的价值越来越重要，数据库系统的安全性成为一个越来越值得关注的方面。

数据库的安全性是指在信息系统的不同层次保护数据库，防止未授权的数据访问，避免数据的泄露、不合法的修改或对数据的破坏。安全性问题不是数据库系统所独有的，它来自各个方面，其中既有数据库本身的安全机制，如用户认证、存取权限、视图隔离、跟踪与审查、数据加密、数据完整性控制、数据访问的并发控制、数据库的备份和恢复等方面，也涉及计算机硬件系统、计算机网络系统、操作系统、组件、Web 服务、客户端应用程序、网络浏览器等。只是在数据库系统中大量数据集中存放，而且为许多最终用户直接共享，从而使安全性问题更为突出，每一个方面产生的安全问题都可能导致数据库数据的泄露、意外修改、丢失等后果。

在一般计算机系统中，安全措施往往是一级一级层层设置的，其安全模型如图 13.1 所示。

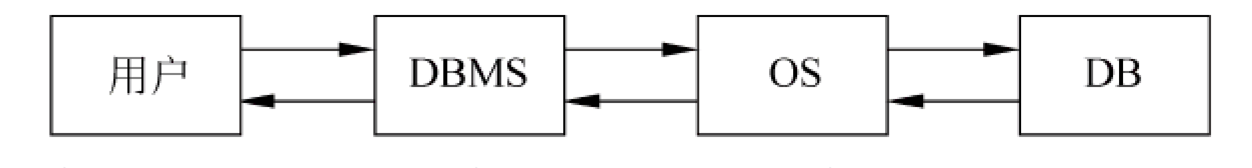


图 13.1　计算机系统的安全模型

在这个安全模型中，用户要求进入计算机时，系统首先根据输入的用户标识进行用户身份鉴定，只有合法用户才准许进入计算机系统。对已进入系统的用户，DBMS 还要设置很多访问限制，例如 DBMS 级访问控制主要有自由存取控制(Discretionary Access Control，DAC)和强制存取控制方法(Mandatory Access Control，MAC)，并只允许用户进行合法操作。操作系统一级也有自己的保护措施，它主要是基于用户访问权限的访问控制。数据最后还可以以密码形式存储到数据库中。

13.1.2　安全层次简介

在安全问题上，DBMS 应与操作系统达到某种意向，理清关系，分工协作，以加强 DBMS 的安全性。数据库系统安全保护措施是否有效是数据库系统的主要指标之一。主要包括以下几个层次的安全措施。

1. 数据库系统层次

数据库管理系统的安全保护主要表现在对数据库的存取权限控制上。同时，数据库本身的完整性问题也直接关系到数据库数据的安全可靠。

2. 操作系统层次

操作系统的安全保护主要表现在标识、鉴别、审核用户以及隔离用户进程。

3. 网络层次

网络层次的安全性包括保密性、安全协议设计、接入控制等。

4. 物理层次

物理层次的安全性主要指物理结点保护、硬件保护等方面所采取的相应措施。

5. 人员层次

在人员层次上，主要采取用户分类、角色设定、授权等措施来防止操作人员对系统的非法访问。

13.1.3　安全标准简介

信息技术和网络空间给社会各个方面都注入了新的活力。人们在享受信息化带来的众多好处的同时，也面临着日益突出的信息安全与保密的问题。越来越多的人开始关注信息安全评估，而评估工作必须依据一定的安全标准。在一系列的安全标准中，最有影响的当推 TCSEC 和 CC 这两个标准。

1. TCSEC 标准

TCSEC 是指 1985 年美国国防部正式颁布的《DoD 可信计算机系统评估准则》。在 TCSEC 中，美国国防部按处理信息的等级和应采用的响应措施，将计算机安全从高到低分为：A、B、C、D 四类八个级别，共 27 条评估准则。随着安全等级的提高，系统的可信度随之增加，风险逐渐减少，如表 13.1 所示。

表 13.1　TCSEC 的等级划分

等级划分	等级名称	保护等级
D 类	最低保护等级	D 级：无保护级
C 类	自主保护级	C1 级：自主安全保护级
		C2 级：控制访问保护级

续表

等级划分	等级名称	保护等级
B类	强制保护级	B1级：标记安全保护级
		B2级：结构化保护级
		B3级：安全区域保护级
A类	验证保护级别	A1级：验证设计级
		超A1级

2. CC(Common Criteria)标准

国际《信息技术安全性评估通用准则》(简称为《通用准则》,CC)是北美和欧盟联合以开发一个统一的国际互认的安全标准的结果,是在美国、加拿大、欧洲等国家和地区分别自行推出的评估标准及具体实践的基础上,通过相互间的总结和互补发展起来的。

3. 我国的信息安全评估标准

我国在信息系统安全的研究与应用方面与其他先进国家相比有一定的差距,但近年来,国内的研究人员已经在安全操作系统、安全数据库、安全网关、防火墙、入侵检测系统等方面做了许多工作,1999年发布的国家强制性标准《计算机信息系统安全保护等级划分准则》(GB17859-1999)为安全产品的研制提供了技术支持,也为安全系统的建设和管理提供了技术指导。该标准是我国计算机信息系统安全保护等级系列标准的第一部分,其他相关应用指南、评估准则等正在建设之中,该标准的制定参照了美国的TCSEC标准。

13.2 Oracle 的安全机制

13.2.1 用户管理

在Oracle中,最外层的安全措施就是让用户标识自己的名字,然后由系统进行核实。Oracle允许用户重复标识三次,如果三次未通过,系统就自动退出。

在Oracle数据库系统中可以通过设置用户的安全参数维护安全性。为了防止非授权用户对数据库进行存取,在创建用户时必须使用安全参数对用户进行限制。用户的安全参数包括:用户名、口令、用户默认表空间、用户临时表空间、用户空间存取限制和用户资源存取限制。

1. 创建用户

当安装和建立Oracle时,系统会自动建立特权用户SYS、管理用户SYSTEM与普通用户SCOTT。同时也可以使用CREATE USER命令来创建一个新的数据库用户,但是创建者必须具有CREATE USER系统权限。

使用SQL命令创建用户的语法如下:

```
CREATE USER 用户名
```

```
IDENTIFIED BY 口令
[DEFAULT TABLESPACE 表空间名]
[TEMPORARY TABLESPACE 表空间名]
[PASSWORD EXPIRE]
[ACCOUNT {LOCK | UNLOCK}]
```

说明：

（1）使用 IDENTIFIED BY 子句为用户设置口令，这时用户将通过数据库来进行身份认证。如果要通过操作系统来对用户进行身份认证，则必须使用 IDENTIFIED EXTERNAL BY 子句。

（2）使用 DEFAULT TABLESPACE 子句为用户指定默认表空间。如果没有指定默认表空间，Oracle 会把 SYSTEM 表空间作为用户的默认表空间。

（3）TEMPORARY TABLESPACE 子句：为用户指定临时表空间，若没有指定，Temp 为该用户的临时表空间。

（4）PASSWORD EXPIRE 子句：设置用户口令的初始状态为过期。

（5）ACCOUNT LOCK 子句：设置用户账户的初始状态为锁定，缺省为：ACCOUNT UNLOCK。

例题 13.1　创建一个 test 用户，密码为 test。该用户口令没有到期，账号也没有被锁定，默认表空间为 users，在该表空间的配额为 20MB，临时表空间为 temp。

例题解析：

```
CREATE USER test
IDENTIFIED BY test
DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp
QUOTA 20M ON users
ACCOUNT UNLOCK;
```

说明：当创建用户后，必须给该用户授权，用户才能连接到数据库，并对数据库中的对象进行操作。只有拥有 CREATE SESSION 权限的用户才能连接到数据库。可用下列语句对 test 用户授权。

```
GRANT CREATE SESSION TO test;
```

2. 修改用户

建立用户时指定的所有特性都可以使用 ALTER USER 命令加以修改。

语法如下：

```
ALTER USER 用户名
IDENTIFIED  BY 口令
[DEFAULT TABLESPACE 表空间名]
[TEMPORARY TABLESPACE 表空间名]
```

```
[PASSWORD EXPIRE]
[ACCOUNT {LOCK | UNLOCK}]
```

例题 13.2 将 test 用户的口令修改为 tiger，并且将其口令设置为到期。

例题解析：

```
ALTER USER test
IDENTIFIED BY tiger
PASSWORD EXPIRE;
```

例题 13.3 修改 test 用户的默认表空间和账户的状态。将默认表空间改为 system，账户的状态设置为锁定状态。

例题解析：

```
ALTER USER test
DEFAULT TABLESPACE system
ACCOUNT LOCK;
```

说明：修改用户的默认表空间只影响将来建立的对象，以前建立的对象仍然存放在原来的表空间上，将来建立的对象放到新的默认空间。

3. 删除用户

使用 DROP USER 命令可以从数据库中删除一个用户。假如用户拥有对象，必须指定 CASCADE 关键字才能删除用户，否则返回一个错误。假如指定了 CASCADE 关键字，Oracle 先删除该用户所拥有的所有对象，然后删除该用户。

语法如下：

```
DROP USER 用户名;
```

例题 13.4 删除 test 用户。

例题解析：

```
DROP USER test;
```

说明：如果我们已经在 test 用户下创建了相应的对象，如表、视图，那么我们在使用上述命令对用户进行删除时将出现错误，此时语句应改为"DROP USER test CASCADE;"，但是，一个连接到 Oracle 服务器的用户是不能被删除的。

4. 查询用户信息

可以通过查询数据字典视图或动态性能视图来获取用户信息。

(1) ALL_USERS：包含数据库所有用户的用户名、用户 ID 和用户创建时间。

(2) DBA_USERS：包含数据库所有用户的详细信息。

(3) USER_USERS：包含当前用户的详细信息。

(4) V$SESSION：包含用户会话信息。

(5) V$OPEN_CURSOR：包含用户执行的 SQL 语句信息。

普通用户只能查询 USER_USERS 数据字典，只有拥有 DBA 权限的用户才能查询 DBA_USERS 数据字典。

例题 13.5　查询当前用户的详细信息。

例题解析：

```
SELECT USERNAME,
DEFAULT_TABLESPACE,
TEMPORARY_TABLESPACE,
ACCOUNT_STATUS, EXPIRY_DATE
FROM USER_USERS;
```

程序运行效果如图 13.2 所示。

```
USERNAME                          DEFAULT_TABLESPACE
------------------------------    ------------------------------
TEMPORARY_TABLESPACE              ACCOUNT_STATUS                    EXPIRY_DATE
------------------------------    --------------------------------  -----------
SYSTEM                            SYSTEM
TEMP                              OPEN                              25-3月 -15
```

图 13.2　当前用户的详细信息

例题 13.6　查询数据库中所有用户名、默认表空间和账户的状态。

例题解析：

```
SELECT USERNAME,
DEFAULT_TABLESPACE,
ACCOUNT_STATUS
FROM DBA_USERS;
```

程序运行效果如图 13.3 所示。

13.2.2　权限管理

在 Oracle 中，权限分为两类：系统权限和对象权限。

1. 系统权限

系统权限是指在系统级控制数据库的存取和使用的机制，系统权限决定了用户是否可以连接到数据库以及在数据库中可以进行哪些操作。

系统权限的分类可划分成下列三类：

允许在系统范围内操作的权限。如 CREATE SESSION、CREATE TABLESPACE 等与用户无关的权限。

允许在用户自己的账号内管理对象的权限。如 CREATE TABLE 等建立、修改、删除指定对象的权限。

```
USERNAME                       DEFAULT_TABLESPACE             ACCOUNT_STATUS
------------------------------ ------------------------------ ----------------
MGMT_VIEW                      SYSTEM                         OPEN
SYS                            SYSTEM                         OPEN
SYSTEM                         SYSTEM                         OPEN
DBSNMP                         SYSAUX                         OPEN
SYSMAN                         SYSAUX                         OPEN
OUTLN                          SYSTEM                         EXPIRED & LOCKED
FLOWS_FILES                    SYSAUX                         EXPIRED & LOCKED
MDSYS                          SYSAUX                         EXPIRED & LOCKED
ORDSYS                         SYSAUX                         EXPIRED & LOCKED
EXFSYS                         SYSAUX                         EXPIRED & LOCKED
WMSYS                          SYSAUX                         EXPIRED & LOCKED
APPQOSSYS                      SYSAUX                         EXPIRED & LOCKED
APEX_030200                    SYSAUX                         EXPIRED & LOCKED
OWBSYS_AUDIT                   SYSAUX                         EXPIRED & LOCKED
ORDDATA                        SYSAUX                         EXPIRED & LOCKED
CTXSYS                         SYSAUX                         EXPIRED & LOCKED
ANONYMOUS                      SYSAUX                         EXPIRED & LOCKED
XDB                            SYSAUX                         EXPIRED & LOCKED
ORDPLUGINS                     SYSAUX                         EXPIRED & LOCKED
OWBSYS                         SYSAUX                         EXPIRED & LOCKED
SI_INFORMTN_SCHEMA             SYSAUX                         EXPIRED & LOCKED
OLAPSYS                        SYSAUX                         EXPIRED & LOCKED
SCOTT                          USERS                          EXPIRED & LOCKED
ORACLE_OCM                     USERS                          EXPIRED & LOCKED
XS$NULL                        USERS                          EXPIRED & LOCKED
BI                             USERS                          EXPIRED & LOCKED
PM                             USERS                          EXPIRED & LOCKED
MDDATA                         USERS                          EXPIRED & LOCKED
IX                             USERS                          EXPIRED & LOCKED
SH                             USERS                          EXPIRED & LOCKED
DIP                            USERS                          EXPIRED & LOCKED
OE                             USERS                          EXPIRED & LOCKED
APEX_PUBLIC_USER               USERS                          EXPIRED & LOCKED
HR                             USERS                          EXPIRED & LOCKED
SPATIAL_CSW_ADMIN_USR          USERS                          EXPIRED & LOCKED
SPATIAL_WFS_ADMIN_USR          USERS                          EXPIRED & LOCKED

已选择36行。
```

图 13.3　所有用户的相关信息

允许在任何用户账号内管理对象的权限。如 CREATE ANY TABLE 等带 ANY 的权限，允许用户在任何用户账号下建表。

1）系统权限的授权

使用 GRANT 命令可以将系统权限授予给一个用户、角色或 PUBLIC。

语法如下：

```
GRANT {系统权限|角色} [,{系统权限|角色}]…
TO {用户|角色| PUBLIC} [,{用户|角色| PUBLIC}]…
[WITH ADMIN OPTION]
```

说明：

（1）PUBLIC 是创建数据库时自动创建的一个特殊的用户组，数据库中所有的用户都属于该用户组。

（2）WITH ADMIN OPTION 表示允许得到权限的用户进一步将这些权限或角色授予

给其他的用户或角色。

例题 13.7　先创建用户 user1，再为用户 user1 授予 CREATE SESSION 系统权限，保证用户 user1 成功登录。

例题解析：

创建用户 user1，结果如图 13.4 所示。

```
SQL> CREATE USER user1
  2  IDENTIFIED BY user1
  3  DEFAULT TABLESPACE users
  4  TEMPORARY TABLESPACE temp
  5  QUOTA 20M ON users
  6  ACCOUNT UNLOCK;

用户已创建。
```

图 13.4　用户 user1 的创建

此时登录时，系统会拒绝并给出如图 13.5 所示的提示信息。

```
SQL> conn user1
输入口令: *****
ERROR:
ORA-01045: user USER1 lacks CREATE SESSION privilege; logon denied


警告: 您不再连接到 ORACLE。
```

图 13.5　用户 user1 登录失败

通过授权解决用户登录失败的问题，程序如下：

```
GRANT CREATE SESSION
TO user1;
```

为用户 user1 授予权限后，再次登录的结果如图 13.6 所示。

```
SQL> conn user1;
输入口令: *****
已连接。
```

图 13.6　用户 user1 登录成功

例题 13.8　为用户 user1 授予 CREATE TABLE 系统权限。

例题解析：

在例题 13.7 中用户 user1 已经创建成功，并且能够成功登录数据库。在用户 user1 中创建一张基本表，结果如图 13.7 所示。

当前用户 user1 不具有创建表的权限，所以创建失败了。在 SYSTEM 用户下，给用户 user1 授权，程序如下：

```
GRANT CREATE TABLE
TO user1;
```

```
SQL> CREATE TABLE student
  2  (sno CHAR(8),
  3   sname VARCHAR2(20),
  4   age NUMBER);
CREATE TABLE student
*
第 1 行出现错误:
ORA-01031: 权限不足
```

图 13.7　创建基本表失败

为用户 user1 授予权限后，在用户 user1 中再次创建基本表，结果如图 13.8 所示。

```
SQL> CREATE TABLE student
  2  (sno CHAR(8),
  3   sname VARCHAR2(20),
  4   age NUMBER);

表已创建。
```

图 13.8　创建基本表成功

例题 13.9　为用户 user1 授予 CREATE VIEW 系统权限，允许用户 user1 将该权限再授予给其他用户。

例题解析：

```
GRANT CREATE VIEW
TO user1
WITH ADMIN OPTION;
```

2）系统权限的回收

使用 REVOKE 命令可以从用户或角色上回收系统权限。

语法如下：

```
REVOKE {系统权限|角色} [,{系统权限|角色}]…
FROM {用户名|角色| PUBLIC} [,{用户名|角色| PUBLIC}]…
```

说明：

（1）多个管理员授予用户同一个系统权限后，其中一个管理员回收其授予该用户的系统权限时，不影响该用户从其他管理员处获得系统权限。

（2）系统权限授权语句中 WITH ADMIN OPTION 从句给了授权者将此权限再授予给另一个用户或 PUBLIC 的权利。但是当一个系统权限回收时没有级联影响，不管在权限授予时是否带 WITH ADMIN OPTION 从句。

例题 13.10　回收用户 user1 的 CREATE VIEW 系统权限。

例题解析：

```
REVOKE CREATE VIEW
```

```
FROM user1;
```

2. 对象权限

对象权限是指在对象级控制数据库的存取和使用的机制，用于设置一个用户对其他用户的表、视图、序列、过程、函数、包的操作权限。对于不同类型的对象，有不同类型的对象权限。对于有些模式对象，如聚集、索引、触发器、数据库链接等没有相关的对象权限，这些权限由系统进行控制。

补充：模式(schema)对象是具有拥有者的对象(如：表 HR.TABLE1 是用户 HR 拥有的，名为 TABLE1 的表)。数据库还包含非模式对象。非模式对象与用户无关，在某些情况下，非模式对象是用户系统拥有的对象，并且可以被所有用户访问，这种非模式对象包括公共的同义词与公共的数据库链接，所有用户不需要考虑任何权限因素就能够使用这些对象。

Oracle 提供的对象权限如表 13.2 所示。

表 13.2　Oracle 提供的对象权限

对象权限＼对象	TABLE	COLUMN	VIEW	SEQUENCE	PROCEDURE/FUNCTION/PACKAGE
ALTER	√			√	
DELETE	√		√		
EXECUTE					√
INDEX	√				
INSERT	√	√	√		
REFERENCES	√	√			
SELECT	√		√	√	
UPDATE	√	√	√		
READ					

1) 授权

使用 GRANT 命令可以将对象权限授予给一个用户、角色或 PUBLIC。

语法如下：

```
GRANT {对象权限[(列名 1 [,列名 2… ])]
[,对象权限[(列名 1 [,列名 2… ])]]… |ALL}
ON 对象名
TO {用户名|角色名| PUBLIC} [,{用户名|角色名| PUBLIC}]…
[WITH GRANT OPTION]
```

说明：

WITH GRANT OPTION 表示允许得到权限的用户进一步将这些权限授予给其他的用户或角色。

例题 13.11 用户 system 将学生表 student 的 SELECT 权限和属性列 sname、age 上的 UPDATE 权限授予给用户 user1，并且允许用户 user1 再将这些对象权限授予给其他用户。

例题解析：

授权前，用户 user1 的权限测试结果如图 13.9 所示。

```
SQL> conn user1
输入口令:
已连接。
SQL> SELECT * FROM system.student WHERE sno='11432003';
SELECT * FROM system.student WHERE sno='11432003'
                                    *
第 1 行出现错误:
ORA-00942: 表或视图不存在


SQL> UPDATE system.student SET age=age+1 WHERE sno='11432003';
UPDATE system.student SET age=age+1 WHERE sno='11432003'
              *
第 1 行出现错误:
ORA-00942: 表或视图不存在
```

图 13.9 授权前，用户 user1 的权限测试

通过给用户 user1 授权，解决上述问题，程序如下：

```
GRANT SELECT,UPDATE(sname,age)
ON student
TO user1
WITH GRANT OPTION;
```

授权后，用户 user1 的权限测试结果如图 13.10 所示。

```
SQL> conn user1
输入口令:
已连接。
SQL> SELECT * FROM system.student WHERE sno='11432003';

SNO      SNAME                SEX         AGE DEPT
-------- -------------------- ---- ---------- ----------------
11432003 张三                 女           20 侦查系

SQL> UPDATE system.student SET age=age+1 WHERE sno='11432003';

已更新 1 行。
```

图 13.10 授权后，用户 user1 的权限测试

说明：假如用户拥有了一个对象，他就自动地获得了该对象的所有权限。对象拥有者可以将自己对象的操作权授予给别人。例如，用户 system 可以将 SELECT，INSERT，UPDATE，DELETE 等权限授予给其他用户。

2）回收权限

通过使用 REVOKE 命令可以实现权限的回收。

语法如下：

```
REVOKE {用户权限|角色} [,{用户权限|角色}]…
FROM {用户名|角色| PUBLIC} [,{用户名|角色| PUBLIC}]…
[RESTRICT | CASCADE]
```

说明：

(1) 可选项[RESTRICT | CASCADE]中，CASCADE 表示回收权限时要引起级联回收。即从用户 A 回收权限时，要把用户 A 转授出去的同样的权限同时回收。RESTRICT 表示，当不存在级联连锁回收时，才能回收权限，否则系统拒绝回收。

(2) 当使用 WITH GRANT OPTION 从句授予对象权限时，一个对象权限回收时存在级联影响。

例题 13.12　用户 system 从用户 user1 中回收学生表 student 上 SELECT 权限。

例题解析：

回收权限前，测试用户 user1 的权限结果如图 13.11 所示。

```
SQL> show user;
USER 为 "USER1"
SQL> SELECT * FROM system.student WHERE sno='11432001';

SNO      SNAME                SEX         AGE DEPT
-------- -------------------- ---- ---------- ----------
11432001 陈一                 男           18 侦查系
```

图 13.11　回收权限前，测试用户 user1 的权限

用户 system 从用户 user1 中回收学生表 student 上 SELECT 权限，程序如下：

```
REVOKE SELECT
ON student
FROM user1;
```

回收权限后，测试用户 user1 的权限结果如图 13.12 所示。

```
SQL> show user;
USER 为 "USER1"
SQL> SELECT * FROM system.student WHERE sno='11432001';
SELECT * FROM system.student WHERE sno='11432001'
                                        *
第 1 行出现错误:
ORA-01031: 权限不足
```

图 13.12　回收权限后，测试用户 user1 的权限

说明：多个管理员授予用户同一个对象权限后，其中一个管理员回收其授予该用户的对象权限时，不影响该用户从其他管理员处获得的对象权限。如果一个用户获得的对象权

限具有传递性(授权时使用了 WITH GRANT OPTION 子句),并且给其他用户授权,那么该用户的对象权限被回收后,其他用户的对象权限也被回收。

3. 查询各种权限

可以通过数据字典视图查询数据库相应权限信息。对象权限有关的数据字典视图如表 13.3 所示。

表 13.3 对象权限有关的数据字典视图

数据字典视图	描　　述
DBA_TAB_PRIVS	包含数据库所有对象的授权信息
ALL_TAB_PRIVS	包含数据库所有用户和 PUBLIC 用户组的对象授权信息
USER_TAB_PRIVS	包含当前用户对象的授权信息
DBA_COL_PRIVS	包含所有字段已授予的对象权限
ALL_COL_PRIVS	包含所有字段已授予的对象权限信息
USER_COL_PRIVS	包含当前用户所有字段已授予的对象权限信息
DBA_SYS_PRIVS	包含授予用户或角色的系统权限信息
USER_SYS_PRIVS	包含授予当前用户的系统权限信息

例题 13.13 查询当前用户 system 所具有的权限。

例题解析:

```
SELECT username,privilege,admin_option FROM user_sys_privs;
```

程序运行结果如图 13.13 所示。

```
USERNAME                       PRIVILEGE                                ADM
------------------------------ ---------------------------------------- ---
SYSTEM                         GLOBAL QUERY REWRITE                     NO
SYSTEM                         CREATE MATERIALIZED VIEW                 NO
SYSTEM                         CREATE TABLE                             NO
SYSTEM                         UNLIMITED TABLESPACE                     YES
SYSTEM                         SELECT ANY TABLE                         NO
```

图 13.13 用户 system 所具有的权限

13.2.3 角色管理

Oracle 中的角色可以分为预定义角色和自定义角色两类。当数据库创建时,会自动为数据库预定义一些角色,这些角色主要用来限制数据库管理系统权限。此外,用户也可以根据自己的需求,将一些权限集中到一起,建立用户自定义的角色。

1. 预定义角色

预定义角色的细节可以从 DBA_SYS_PRIVS 数据字典视图中查询到。表 13.4 列出了几个常见的预定义角色。

表 13.4　Oracle 11g 常见的预定义角色

角　色　名	描　　述
CONNECT	连接到数据库的权限，建立数据库链路、序列生成器、同义词、表、视图以及修改会话的权限
RESOURCE	建立表、序列生成器，以及建立过程、函数、包、数据类型、触发器的权限
DBA	带 WITH ADMIN OPTION 选项的所有系统权限，所有系统权限可以被授予给数据库中其他用户或角色 DBA 角色拥有最高级别的权限

2. 用户自定义角色

1）创建角色

使用 CREATE ROLE 命令可以建立角色，角色是属于整个数据库，而不属于任何用户的。当建立一个角色时，该角色没有相关的权限，系统管理员必须将合适的权限授予给角色。此时，角色才是一组权限的集合。

语法如下：

```
CREATE ROLE 角色名[NOT IDENTIFIED | IDENTIFIED {BY 口令}]
```

例题 13.14　建立一个带口令 tiger 的角色 student_role。

例题解析：

```
CREATE ROLE student_role IDENTIFIED BY tiger;
```

2）修改角色

语法如下：

```
ALTER ROLE role_name [ NOT IDENTIFIED ][ IDENTIFIED BY password ] ;
```

使用 ALTER ROLE 命令可以修改角色的口令，但不能修改角色名。

例题 13.15　修改角色 student_role 使其没有口令。

例题解析：

```
ALTER ROLE student_role NOT IDENTIFIED;
```

3）授予角色权限

建立完角色后需要给角色授权，授权后的角色才是一组权限的集合。在数据库运行过程中，可以为角色增加权限，也可以回收其权限。

例题 13.16　将用户 system 中学生表 student 的 SELECT、UPDATE 和 DELETE 权限的集合授予给角色 student_role。

例题解析：

```
GRANT SELECT,UPDATE,DELETE
ON student
```

```
TO student_role;
```

3. 给用户或角色授予角色

可以使用 GRANT 语句将角色授予用户或其他角色。

语法如下：

```
GRANT role_list TO user_list|role_list;
```

例题 13.17 将角色 student_role 授予给用户 user1。

例题解析：

角色授予前，用户 user1 的权限测试结果如图 13.14 所示。

```
SQL> show user;
USER 为 "USER1"
SQL> SELECT * FROM system.student WHERE dept='公安信息系';
SELECT * FROM system.student WHERE dept='公安信息系'
                    *
第 1 行出现错误:
ORA-01031: 权限不足
```

图 13.14 角色授予前，用户 user1 的权限测试

将角色 student_role 授予给用户 user1 后，用户 user1 就具有了相应的权限，程序如下：

```
GRANT student_role TO user1;
```

授权后，测试结果如图 13.15 所示。

```
SQL> show user;
USER 为 "USER1"
SQL> SELECT * FROM system.student WHERE dept='公安信息系';

SNO      SNAME                SEX         AGE DEPT
-------- -------------------- ---- ---------- ------------
11432007 陈七                 女           23 公安信息系
11432008 刘八                 男           21 公安信息系
```

图 13.15 角色授予后，用户 user1 的权限测试

4. 从用户或角色回收角色

可以使用 REVOKE 语句从用户或其他角色回收角色。

语法如下：

```
REVOKE role_list FROM user_list|role_list;
```

例题 13.18 将角色 student_role 从用户 user1 回收。

例题解析：

角色回收前，用户 user1 的权限测试结果如图 13.16 所示。

将角色 student_role 从用户 user1 中回收，程序如下：

```
SQL> show user;
USER 为 "USER1"
SQL> SELECT * FROM system.student WHERE age<19;

SNO      SNAME                SEX         AGE DEPT
-------- -------------------- ---- ---------- ----------
11432001 陈一                 男           17 侦查系
11432009 张九                 女           18 治安管理系
```

图 13.16 角色回收前,用户 user1 的权限测试

```
REVOKE student_role FROM user1;
```

角色回收后,用户 user1 就失去了相应的权限,测试结果如图 13.17 所示。

```
SQL> show user;
USER 为 "USER1"
SQL> SELECT * FROM system.student WHERE age<19;
SELECT * FROM system.student WHERE age<19
                            *
第 1 行出现错误:
ORA-01031: 权限不足
```

图 13.17 角色回收后,用户 user1 的权限测试

5. 删除角色

使用 DROP ROLE 命令可以删除角色。即使此角色已经被授予给一个用户,数据库也允许用户删除该角色。

例题 13.19 从数据库中删除 student_role 角色。

例题解析:

```
DROP ROLE student_role;
```

6. 查询角色信息

可以通过数据字典视图或动态性能视图获取数据库角色相关信息。与角色有关的数据字典视图如表 13.5 所示。

表 13.5 与角色有关的数据字典视图

数据字典视图	描　述
DBA_ROLES	包含数据库中所有角色及其描述
DBA_ROLE_PRIVS	包含为数据库中所有用户和角色授予的角色信息
USER_ROLE_PRIVS	包含为当前用户授予的角色信息
ROLE_ROLE_PRIVS	为角色授予的角色信息
ROLE_SYS_PRIVS	为角色授予的系统权限信息
ROLE_TAB_PRIVS	为角色授予的对象权限信息
SESSION_PRIVS	当前会话所具有的系统权限信息

例题 13.20 查询当前用户 system 所具有的角色。

例题解析：

```
SELECT * FROM user_role_privs;
```

程序运行效果如图 13.18 所示。

```
USERNAME                       GRANTED_ROLE                   ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
SYSTEM                         AQ_ADMINISTRATOR_ROLE          YES YES NO
SYSTEM                         DBA                            YES YES NO
SYSTEM                         MGMT_USER                      NO  YES NO
```

图 13.18 用户 system 拥有的角色

例题 13.21 查询角色 exp_full_database 所拥有的权限。

例题解析：

```
SELECT * FROM role_sys_privs WHERE role = 'EXP_FULL_DATABASE';
```

程序运行效果如图 13.19 所示。

```
ROLE                           PRIVILEGE                                ADM
------------------------------ ---------------------------------------- ---
EXP_FULL_DATABASE              READ ANY FILE GROUP                      NO
EXP_FULL_DATABASE              EXECUTE ANY PROCEDURE                    NO
EXP_FULL_DATABASE              RESUMABLE                                NO
EXP_FULL_DATABASE              SELECT ANY TABLE                         NO
EXP_FULL_DATABASE              EXECUTE ANY TYPE                         NO
EXP_FULL_DATABASE              CREATE SESSION                           NO
EXP_FULL_DATABASE              BACKUP ANY TABLE                         NO
EXP_FULL_DATABASE              ADMINISTER RESOURCE MANAGER              NO
EXP_FULL_DATABASE              ADMINISTER SQL MANAGEMENT OBJECT         NO
EXP_FULL_DATABASE              SELECT ANY SEQUENCE                      NO
EXP_FULL_DATABASE              CREATE TABLE                             NO

已选择11行。
```

图 13.19 查看角色 exp_full_database 的权限

13.2.4 视图机制

视图是数据库系统提供给用户以多种角度观察数据库中数据的重要机制，是从一个或几个基表(或视图)导出的表，它与基表不同，是一个虚表。数据库中只存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。

从某种意义上讲，视图就像一个窗口，透过它可以看到数据库中自己感兴趣的数据及其变化。进行存取权限控制时，可以为不同的用户定义不同的视图，把访问数据的对象限制在一定的范围内，也就是说，通过视图机制把要保密的数据对无权存取的用户隐藏起来，从而对数据提供一定程度的安全保护。视图的创建与应用已经在第 3 章讲解过，这里就不再介绍了。

13.2.5　审计

Oracle 数据库的审计功能与 SQL Server 相比更加灵活。Oracle 11g 提供了细粒度的审计，允许用户定义审计策略，可以对对象、权限、对象访问、SQL 语句的类型等等进行审计，审计结果被存储在 SYS 用户的数据库字典中，数据库管理员可以查询该字典，从而获取审计结果。还可以开启报警功能，这样管理员可以在出现安全问题时迅速接到通知。

Oracle 数据库中的审计大概可以分为以下几种类型。

1. 语句审计

按照语句类型审计 SQL 语句，而不论访问何种特定的模式对象。也可以在数据库中指定一个或多个用户，针对特定的语句审计这些用户。

2. 权限审计

审计系统权限，例如 CREATE TABLE 或 ALTER INDEX。和语句审计一样，权限审计可以指定一个或多个特定的用户作为审计的目标。

3. 模式对象审计

审计特定模式对象上运行的特定语句(例如，学生表 student 的 UPDATE 语句)。模式对象审计总是应用于数据库中的所有用户。

4. 细粒度的审计

根据访问对象的内容来审计表访问和权限。使用程序包 DBMS_FGA 来建立特定表上的策略。

Oracle 中的 AUDIT 语句用来设置审计功能，NOAUDIT 语句用来取消审计功能。

例题 13.22　对修改学生表 student 的表结构或其中数据的操作进行审计。

```
AUDIT ALTER,UPDATE
ON student;
```

例题 13.23　取消对学生表 student 修改表结构和修改表数据的审计。

```
NOAUDIT ALTER,UPDATE
ON student;
```

13.2.6　数据加密

通过数据加密可以提高存储数据的安全性，Oracle 数据库加密功能的实现由数据库平台提供的软件包来支持。Oracle 提供了特殊 DBMS_OBFUSCATION_TOOLKIT 包、DBMS_CRYPTO 包等用于数据加密/解密，同时支持 DES、AES 等多种加密/解密算法。

在数据加密中，密钥的存储和管理是非常重要的，它直接影响到数据加密的安全性。但是，在数据库管理系统内核层加密策略中，并没有提供密钥存储的方法，这也是在以 Oracle 提供的安全包为基础制定加密策略时最难处理的部分。在制定密钥的存储和管理方案时，

要确保以下两点：

(1) 密钥存储，足够可靠，以确保能够保护数据。

(2) 要保证合法用户且只有合法用户可以获取密钥。

Oracle 数据库文件加密中，密钥仍然是以数据表形式存放在数据库中。如果密钥以明文形式存放在数据库中，那么攻击者只要进入数据库系统中，就能很容易找到破解密文的密钥。如果密钥以密文形式存放在数据库中，那么加密密钥的密钥如何存放就成了需要解决的新问题。常用的解决方法是采用多级密钥存储管理，把用户密钥与数据密钥结合使用，提高数据库加密的安全性。任何加密策略的安全性都依赖于密钥的安全，但是 Oracle 数据库系统提供的加密方案中，并没有给出密钥存储与管理的安全方法。要想提高数据库加密系统的安全性，还需要辅助其他的外部手段来实现密钥的安全存储与管理。

13.3 数据库完整性控制

13.3.1 完整性基本含义

数据库的安全性和完整性是数据库安全保护的两个不同的方面。数据库的安全性保护数据库以防止不合法用户故意造成的破坏，数据库的完整性保护数据库以防止合法用户无意中造成的破坏。从数据库的安全保护角度来讲，完整性和安全性是密切相关的。

数据库的完整性的基本含义是指数据库中数据的正确性、有效性和相容性，其主要目的是防止错误的数据进入数据库。正确性是指数据的合法性，例如数值型数据只能含有数字而不能含有字母。有效性是指数据是否属于所定义域的有效范围。相容性是指表示同一事实的两个数据应当一致，不一致即是不相容。

13.3.2 完整性约束条件

数据库系统是对现实系统的模拟，现实系统中存在各种各样的规章制度，以保证系统正常、有序地运行。许多规章制度可转化为对数据的约束，例如，单位人事制度中对职工的退休年龄会有规定，也可能一个部门的主管不能在其他部门任职、职工工资只能升不能降等。对数据库中的数据设置某些约束机制，这些添加在数据上的语义约束条件称为数据库完整性约束条件，简称“数据库的完整性”。

SQL 中使用了一系列的概念来描述完整性，包括实体完整性、参照完整性和用户定义完整性。这些完整性一般由 SQL 的 DDL 语句来实现，它们作为模式的一部分存入数据字典中。

一般一条完整性规则可以用一个五元组(D,O,A,C,P)来表示。其中：

(1) D(Data)表示约束作用的对象。

(2) O(Operation)触发完整性检查的数据库操作，即当用户发出什么操作请求时需要

检查该完整性规则，是立即检查还是延迟检查。

(3) A(Assertion)表示数据对象必须满足的断言或语义约束，这是规则的主体。

(4) C(Condition)选择 A 作用的数据对象值的谓词。

(5) P(Procedure)违反完整性规则时触发的过程。

13.3.3　完整性控制机制

DBMS 必须提供一种机制来检查数据库中数据的完整性，看其是否满足语义规定的条件，这种机制称为“完整性检查”。为此，数据库管理系统的完整性控制机制应具有三个方面的功能，来防止合法用户在使用数据库时，向数据库注入不合法或不合语义的数据。

(1) 定义功能——提供定义完整性约束条件的机制。

(2) 验证功能——检查用户发出的操作请求是否违背了完整性约束条件。

(3) 处理功能——如果发现用户的操作请求使数据违背了完整性约束条件，则采取一定的动作来保证数据的完整性。

目前，关系数据库系统都提供了定义和检实体完整性、参照完整性和用户定义完整性的功能。违反实体完整性规则和用户定义完整性规则的操作，一般是采用拒绝执行方式进行处理。对于违反参照完整性的操作，并不都是拒绝执行，也可以接受这个操作，同时执行一些附加的操作，以保证数据库的状态正确。

13.4　实　　验

13.4.1　实验 1　用户管理

1. 实验目的

(1) 掌握数据库用户的创建方法。

(2) 掌握数据库用户的修改方法。

(3) 掌握数据库用户的删除方法。

2. 实验内容

(1) 创建一个 test2 用户，密码为 test2。默认表空间为 system，在该表空间的配额为 10MB。使用新创建的用户 test2 登录数据库，如果不能立即登录，出现错误提示信息，请给出理由。

(2) 创建一个 test3 用户，密码为 test3。默认表空间为 users，在该表空间的配额为 20MB，临时表空间为 temp。该用户的口令初始为过期，账户初始设置为锁定状态。

(3) 修改 test3 用户，将密码改为 tiger，默认表空间改为 system，账户状态设置为解锁状态。

(4) 创建一个 test4 用户，密码为 test4。账户初始设置为锁定状态。

（5）删除 test4 用户。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，按照实验步骤完成相应实验内容。用户管理的语句无语法错误，书写规范，运行结果正确为优秀。如果出现错误，则根据错误点数以及难易程度灵活给分。

13.4.2 实验 2 权限管理

1. 实验目的

（1）掌握权限的授予方法。

（2）掌握权限的回收方法。

2. 实验内容

（1）为实验 1 中创建的用户 test2 授予 CREATE SESSION 、CREATE TABLE 和 CREATE VIEW 系统权限，并且允许用户 test2 将相关权限授予给其他用户。

（2）由用户 test2 将 CREATE TABLE 系统权限授予给用户 test3。

（3）回收用户 test2 的 CREATE VIEW 系统权限。

（4）将用户 system 下员工表 emp 的 SELECT 和 INSERT 权限授予给用户 test2。

（5）从用户 test2 处收回对员工表 emp 的 INSERT 权限。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，按照实验步骤完成相应实验内容。权限管理的语句无语法错误，书写规范，运行结果正确为优秀。如果出现错误，则根据错误点数以及难易程度灵活给分。

13.4.3 实验 3 角色管理

1. 实验目的

（1）掌握角色的创建和修改方法。

（2）掌握角色的授予方法。

（3）掌握角色的删除方法。

2. 实验内容

（1）建立一个不带口令的角色 emp_role。

（2）将用户 system 下员工表 emp 的 SELECT 和 UPDATE 权限授予给角色 emp_role。

（3）将角色 emp_role 授权给用户 scott。

（4）从 scott 用户回收 emp_role 角色。

（5）删除角色 emp_role。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，按照实验步骤完成相应实验内容。角色管理的语句无语法错误，书写规范，运行结果正确为优秀。如果出现错误，则根据错误点数以及难易程度灵活给分。

13.5　本章小结

本章首先介绍了数据库安全性的含义、计算机系统的安全模型、五个层次的安全措施和安全标准。

其次介绍了 Oracle 数据库的安全机制，包括用户管理、权限管理、角色管理、视图机制、审计功能和数据加密。

最后介绍了数据库的完整性控制，包括完整性的基本含义、约束条件和控制机制。

13.6　课后习题

一、选择题

1. 以下哪个不是创建用户过程中必要的信息？(　　)

A. 用户名　　B. 用户权限　　C. 临时表空间　　D. 口令

2. SQL 语言的 GRANT 和 REVOKE 语句主要用来进行授权与回收授权，其主要目的是用来维护数据库的(　　)。

A. 完整性　　B. 可靠性　　C. 安全性　　D. 一致性

3. 当对用户授予系统权限时，使用(　　)从句表示允许得到权限的用户进一步将这些权限授予其他的用户。

A. WITH ADMIN OPTION　　B. WITH REVOKE OPTION

C. WITH GRANT OPTION　　D. WITH USER OPTION

4. 下列(　　)权限不是用户权限。

A. SELECT　　B. INSERT　　C. UPDATE　　D. CREATE

5. 关于角色的说法不正确的是(　　)。

A. 将角色授予用户使用 GRANT 命令

B. 角色一旦授予，不能收回

C. 角色是属于整个数据库，而不属于任何用户的。

D. 删除角色，使用 DROP 命令

二、应用题

1. 创建一个用户 test_user，密码为 tiger，用户的默认表空间为 users，该用户的口令没有到期，账户被锁定。

2. 修改用户 test_user,使其账户解锁。

3. 将用户 system 下的课程表 course 的查询权限和删除权限授予给用户 test_user,用户 test_user 同时获得将这些权限转授给其他用户的权限。

4. 将用户 system 下的课程表 course 的课程名称属性列的修改权限授予给用户 test_user。

5. 从用户 test_user 处收回对课程表 course 的删除权限,若用户 test_user 已经把获得的删除权限转授给其他用户,则需要级联收回。

6. 建立一个不带口令的角色 test_role。

7. 将用户 system 下部门表 dept 的查询权限和更新权限授予给角色 test_role。

8. 将角色 test_role 授权给用户 test_user。

9. 从所有用户身上回收角色 test_role。

10. 删除角色 test_role。

11. 删除用户 test_user。

第 14 章

数据库的备份与恢复

14.1 事　　务

14.1.1 事务的定义

事务其实是一个很简单的概念,用户每天都会遇到许多现实生活中类似于事务的示例。例如,商业活动中的交易,对于任何一笔交易来说,都涉及两个基本动作:一手交钱和一手交货。这两个动作构成了一个完整的商业交易,缺一不可。也就是说,这两个动作都成功发生,说明交易完成;如果只发生一个动作,则交易失败。所以,为了保证交易能够正常完成,需要某种方法来保证这些操作的整体性,即这些操作要么都成功,要么都失败。这就是事务在数据库中的作用。

事务是指作为单个逻辑工作单元执行的一系列操作序列,该序列包含了一组数据库操作命令。这组操作命令作为一个整体,要么都执行,要么都不执行。

在关系数据库系统中,一个事务可以是一条 SQL 语句,也可以是一组 SQL 语句。

14.1.2 事务的特性

事务具有四个特性:原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)和持续性(Durability)。这四个特性也简称为 ACID 特性。

1. 原子性

事务是数据库的逻辑工作单位,事务中包括的各操作要么都做,要么都不做。

2. 一致性

事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。因此当数据库中只包含成功事务提交的结果时,就说数据库处于一致性状态。如果数据库系统运行中发生故障,有些事务尚未完成就被迫中断,系统将事务中对数据库的所有已完成的操作全部撤销,回退到事务开始时的一致状态。

3. 隔离性

一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对其他并发事务是隔离的,并发执行的各个事务之间不能互相干扰。

4. 持续性

持续性指一个事务一旦提交,它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其执行结果有任何影响。

事务是恢复和并发控制的基本单位。保证事务 ACID 特性是事务处理的重要任务。事务 ACID 特性可能遭到破坏的因素有:

(1) 多个事务并行运行时,不同事务的操作交叉执行。

(2) 事务在运行过程中被强行停止。

在第一种情况下,数据库管理系统必须保证多个事务的交叉运行不影响这些事务的原子性。在第二种情况下,数据库管理系统必须保证被强行终止的事务对数据库和其他事务没有任何影响。这些就是数据库管理系统中恢复机制和并发控制机制的责任。

14.1.3 事务控制语句

在 Oracle 中,没有提供开始事务处理语句,所有的事务都是隐式开始的。也就是说,在 Oracle 中,用户不可以显式使用命令来开始一个事务。Oracle 认为第一条修改数据库的语句,或者一些要求事务处理的场合都是事务隐式的开始。但是当用户想要终止一个事务处理时,必须显式使用 COMMIT 和 ROLLBACK 语句结束。

根据事务的 ACID 特定,Oracle 提供了如下一组语句对事务进行控制。

1. SET TRANSACTION

设置事务的属性。

2. SET CONSTRAINS

在当前事务中设置约束模式。约束模式是指在事务中修改数据时,数据库中的约束是立即应用于数据,还是将约束推迟到当前事务结束后应用。

3. SAVEPOINT

在事务中建立一个存储点。当事务处理发生异常而回滚事务时,可指定事务回滚到某存储点,然后从该存储点重新执行。

4. RELEASE SAVEPOINT

删除一个存储点。

5. ROLLBACK

回滚事务,即取消对数据库所做的任何修改。

6. COMMIT

提交事务,即把事务中对数据库的修改进行永久保存。

14.2 数据库的恢复技术

尽管数据库系统中采取了各种保护措施来防止数据库的安全性和完整性被破坏,保证并发事务的正确执行,但是计算机系统中硬件的故障(如磁盘损坏、电源故障)、软件的错误、

操作人员的失误以及某些恶意的破坏仍是不可完全避免的。这些故障轻则造成运行事务非正常中断，影响数据库中数据的正确性，重则破坏数据库，使数据库中全部或部分数据丢失。因此，DBMS 必须及时而正确地进行数据库结构、对象和数据的复制，以便在数据库遭到破坏的时候能够修复数据库，这就是数据库的备份；同时 DBMS 必须具有保证事务的原子性及把数据库从错误状态恢复到某个已知的正确状态的功能，这就是数据库的恢复。

14.2.1　故障的种类

数据库系统中可能发生各种各样的故障，破坏事务原子性和引起数据库错误的原因很多，大致可以分为以下 4 类。

1. 事务内部的故障

它是指由于事务没有达到预期的终点，导致数据库可能处于一种不正确的状态。见下面转账事务的例子。

例如，银行转账事务，这个事务把一笔金额从一个账户甲转给另一个账户乙。

```
BEGIN TRANSACTION
读账户甲的余额 BALANCE;
BALANCE = BALANCE - AMOUNT; (AMOUNT 为转账金额)
IF(BALANCE<0)THEN
{
打印'金额不足,不能转账';
ROLLBACK; (撤销刚才的修改,恢复事务); }
ELSE
{
读账户乙的余额 BALANCE1;
BALANCE1 = BALANCE1 + AMOUNT;
写回 BALANCE1;
COMMIT; }
```

这个例子所包括的两个更新操作要么全部完成要么全部不做。否则就会使数据库处于不一致状态，例如只把账户甲的余额减少了而没有把账户乙的余额增加。在这段程序中若产生账户甲余额不足的情况，应用程序可以发现并让事务回滚，撤销已做的修改，恢复数据库到正确状态。

事务故障意味着事务没有达到预期的终点，因此，数据库可能处于不正确状态。恢复程序要在不影响其他事务运行的情况下，强行回滚(ROLLBACK)该事务，即撤销该事务已经做出的任何对数据库的修改，使得该事务好像根本没有启动一样。

2. 系统故障

系统故障(通常称为软故障，Soft Crash)是指在造成系统停止运转的任何事件(如硬件故障、操作系统错误、DBMS 代码错误、突然停电等)的影响下，使正在运行的事务都以非正常的方式终止，从而引起内存信息丢失，但未破坏外存中的数据，致使系统需要重新启动。

这类故障发生时,内存内容,尤其是数据库缓存区(在内存)中的内容将可能丢失,所有运行事务都非正常终止。这时,一些尚未完成的事务的结果可能已送入物理数据库,从而造成数据库可能处于不正确的状态。为保证数据一致性,需要清除这些事务对数据库的所有修改。但由于无法确定哪些事务已更新过数据库,因此,在系统重新启动时,恢复子系统必须强行撤销(UNDO)所有未完成事务。

另外,发生系统故障时,有些已完成的事务可能有一部分甚至全部留在缓冲区,尚未写回到磁盘上的物理数据库中,系统故障将导致这些事务对数据库的修改部分或全部丢失,也会使数据库处于不一致状态,因此应将这些事务已提交的结果重新写入数据库。所以,系统重新启动后,恢复子系统除需要撤销所有未完成事务外,还需要重做(REDO)所有已提交的事务,以将数据库真正恢复到一致状态。

3. 介质故障

系统故障常称为软故障(Soft Crash),介质故障称为硬故障(Hard Crash)。硬故障指外存故障,如磁盘损坏、磁头碰撞、瞬时强磁场干扰等。这类故障将破坏数据库或部分数据库,并影响正在存取这部分数据的所有事务。这类故障比前两类故障发生的可能性小得多,但破坏性最大。

这类故障的恢复需要装入数据库发生介质故障前某个时刻的数据副本,重做自此时开始的所有成功事务,将这些事务已提交的结果重新记入数据库中。

4. 计算机病毒

计算机病毒是一种人为的故障或破坏,是一些恶作剧者研制的一种计算机程序。这种程序与其他程序不同,它像微生物学所称的病毒一样可以繁殖和传播,并造成对计算机系统包括数据库的危害。数据库一旦被破坏,仍要求用恢复技术对数据库加以恢复。

总结各类故障,对数据库的影响有两种可能性:一是数据库本身被破坏;二是数据库没有破坏,但数据可能不正确,这是因为事务的运行是被非正常终止造成的。

14.2.2 恢复的实现技术

备份是指数据库管理员定期或不定期地将数据库部分或全部内容复制到磁带或磁盘上保存起来的过程。当数据库遭到破坏时,可以利用备份进行数据库的恢复。所以备份的目的就是当数据库发生意外时,尽可能地减少数据的丢失。何时进行备份,取决于所能承受数据损失的大小。

数据库若要成功地进行恢复,备份的过程中必然涉及的一个关键问题即如何建立冗余数据。建立冗余数据最常用的技术是数据转储和日志文件。通常在一个数据库系统中,这两种方法是一起使用的。

1. 数据转储

所谓转储即 DBA(数据库管理员)定期地将整个数据库复制到磁带或另一个磁盘上保存起来的过程。这些备用的数据文本称为后备副本或后援副本。

当数据库遭到破坏后可以将后备副本重新装入，但重装后备副本只能将数据库恢复到转储时的状态，要想恢复到故障发生时的状态，必须重新运行自转储以后的所有更新事务。

转储是十分耗费时间和资源的，不能频繁进行。DBA 应该根据数据库使用情况确定一个适当的转储周期。

根据转储的状态不同，转储分为：

(1) 静态转储——在系统中无运行事务时进行的转储操作。即转储操作开始的时刻，数据库处于一致性状态，而转储期间不允许（或不存在）对数据库进行任何存取、修改活动。

(2) 动态转储——指转储期间允许对数据库进行存取或修改。即转储和用户事务可以并发执行。

根据转储的方式不同，转储可分为：

(1) 海量转储——每次转储全部数据库。

(2) 增量转储——只转储上次转储后更新过的数据。

2. 日志文件

日志文件是用来记录事务对数据库的更新操作的文件。不同数据库系统采用的日志文件格式并不完全一样，但是日志的功能是相同的。日志文件在数据库恢复中都起着非常重要的作用。可以用来进行事务故障恢复和系统故障恢复，并协助后备副本进行介质故障恢复。

当数据库毁坏后可重新装入后援副本把数据库恢复到转储结束时刻的正确状态，然后利用日志文件，把已完成的事务进行重做处理，对故障发生时尚未完成的事务进行撤销处理。这样不必重新运行那些已完成的事务程序就可把数据库恢复到故障前某一时刻的正确状态。

14.2.3　恢复策略

恢复数据库是指将数据库从错误描述状态恢复到正确的描述状态。下面简单介绍数据库恢复的策略与方法。

1. 事务故障的恢复

事务故障是指事务在运行至正常终止点前被中止，这时恢复子系统应利用日志文件撤销（UNDO）此事务已对数据库进行的修改。事务故障的恢复是由系统自动完成的，对用户是透明的。

2. 系统故障的恢复

系统故障造成数据库不一致状态的原因有两个：一是未完成事务对数据库的更新可能已写入数据库；二是已提交事务对数据库的更新可能还留在缓冲区，没来得及写入数据库。

因此恢复操作就是要撤销故障发生时未完成的事务，重做已完成的事务。系统故障的恢复是由系统在重新启动时自动完成的，不需要用户干预。

3. 介质故障的恢复

发生介质故障后，磁盘上的物理数据和日志文件被破坏，这是最严重的一种故障，恢复方法是重装数据库，然后重做已完成的事务。

介质故障的恢复需要 DBA 介入。但 DBA 只需要重装最近转储的数据库副本和有关的各日志文件副本，然后执行系统提供的恢复命令即可，具体的恢复操作仍由 DBMS 完成。

4. 建立数据库镜像

随着磁盘容量越来越大，价格越来越便宜，为避免磁盘介质出现故障影响数据库的可用性，许多数据库管理系统提供了数据库镜像(Mirror)功能用于数据库恢复。即根据 DBA 的要求，自动把整个数据库或其中的关键数据复制到另一个磁盘上。每当主数据库更新时，DBMS 自动把更新后的数据复制过去，即 DBMS 自动保证镜像数据与主数据的一致性。这样，一旦出现介质故障，可由镜像磁盘继续提供使用，同时 DBMS 自动利用镜像磁盘数据进行数据库的恢复，而不需要关闭系统和重装数据库副本。在没有出现故障时，数据库镜像还可以用于并发操作，即当一个用户对数据加排他锁修改数据时，其他用户可以读镜像数据库上的数据，而不必等待该用户释放锁。

由于数据库镜像是通过复制数据实现的，频繁地复制数据自然会降低系统运行效率，因此在实际应用中用户往往只选择对关键数据和日志文件镜像，而不是对整个数据库进行镜像。

14.3 Oracle 数据库的备份

备份和恢复是两个相互联系的概念，备份是将数据信息保存起来；而恢复则是当意外事件发生或者某种需要时，将已备份的数据信息还原到数据库系统中去。Oracle 数据库的备份方法分为物理备份和逻辑备份。

14.3.1 物理备份

物理备份是针对组成数据库的物理文件的备份。这是一种常用的备份方法，通常按照预定的时间间隔进行。物理备份通常有两种方式：冷备份与热备份。

1. 冷备份

冷备份是指在数据库关闭的情况下将组成数据库的所有物理文件全部备份到磁盘或磁带。冷备份又分为归档模式和非归档模式下的冷备份。

例题 14.1 非归档模式下的冷备份。

例题解析：

(1) 启动 SQL * Plus，以 SYS 身份登录。

(2) 关闭数据库。

```
SQL> SHUTDOWN IMMEDIATE;
```

(3) 复制以下物理文件到相应的磁盘。

所有控制文件、所有数据文件、所有重做日志文件、初始化参数文件。

(4) 重新启动数据库。

```
SQL> STARTUP;
```

例题 14.2　归档模式下的冷备份。

例题解析：

(1) 查看当前存档模式。

```
SQL> ARCHIVE LOG LIST;
```

(2) 修改归档日志存放路径，强制为归档日志设置存储路径。

```
SQL> ALTER system SET log_archive_dest_10 = 'location = e:/orcl';
```

(3) 关闭数据库。

```
SQL> SHUTDOWN IMMEDITE;
```

(4) 启动数据 mount 状态。

```
SQL> STARTUP MOUNT;
```

(5) 修改数据库为归档模式。

```
SQL> ALTER DATABASE ARCHIVELOG;
```

(6) 修改数据库状态。

```
SQL> ALTER DATABASE OPEN;
```

(7) 按上述步骤设置数据库的归档模式，并运行在自动归档模式下。然后进行日志切换，有几个日志文件组，便要日志切换几次，以便将所有日志信息都存储到归档文件。

```
SQL> CONNECT  /AS  SYSDBA;
SQL> ALTER system SWITCH logfile;
SQL> ALTER system SWITCH logfile;
SQL> ALTER system SWITCH logfile;
```

(8) 接着关闭数据库，然后将组成数据库的所有物理文件（包括控制文件、数据文件、重做日志文件）进行完全备份，备份到 e:\orcl\cold\目录下。将归档日志文件也备份到 f:\oracle\arch\目录下。备份完成后重新打开数据库即可。

2. 热备份

热备份又可称为联机备份或 ARCHIVELOG 备份。是指在数据库打开的情况下将组成数据库的控制文件、数据文件备份到磁盘或磁带，当然必须将归档日志文件也一起备份。

热备份要求数据库必须运行在归档模式。

例题 14.3 归档模式下的热备份。

例题解析：

(1) 确保数据库和监听进程已正常启动。

(2) 确保数据库运行在归档模式。

(3) 查询数据字典确认 system、users 表空间所对应的数据文件。

```
SQL> CONNECT  /AS  SYSDBA;
SQL> SELECT file_name,tablespace_name FROM dba_data_files;
```

(4) 将 SYSTEM 表空间联机备份。

因为 system 表空间中存放数据字典信息，所以 system 表空间不能脱机，只能进行联机备份。

```
SQL> ALTER tablespace system BEGIN BACKUP;
SQL> HOST COPY d:\app\Administrator\oradata\orcl\SYSTEM01.DBF e:\orcl\hot\;
SQL> ALTER  tablespace  system  END  BACKUP;
```

(5) 将 users 表空间脱机备份。

非 system 表空间可以进行联机备份，也可以进行脱机备份。users 表空间对应的数据文件有三个。

(6) 数据库中其他表空间都可以用与 users 表空间相同的方法进行联机或脱机备份。

(7) 将当前联机重做日志文件归档。

将当前联机重做日志文件存储为归档日志文件，以便以后恢复时使用。

```
SQL> ALTER  system  ARCHIVE  log  current;
```

或者切换所有的联机日志文件。

```
SQL> ALTER system SWITCH logfile;
SQL> ALTER system SWITCH logfile;
SQL> ALTER system SWITCH logfile;
```

(8) 将控制文件备份。

用下列命令备份控制文件，产生一个二进制文件副本，放在相应目录下。

```
SQL> ALTER  DATABASE  BACKUP
 controlfile to 'e:\orcl\hot\CONTROL01.CTL';
```

14.3.2 逻辑备份

逻辑备份是用 Oracle 系统提供的 EXPORT 工具将组成数据库的逻辑单元(表、用户、数据库)进行备份，将这些逻辑单元的内容存储到一个专门的操作系统文件中。

Oracle 实用工具 EXPORT 利用 SQL 语句读出数据库数据，并在操作系统层将数据和定义存入二进制文件。可以选择导出整个数据库、指定用户或指定表。在导出期间，还可以选择是否导出与表相关的数据字典的信息，如权限、索引和与其相关的约束条件。导出共有三种模式，具体介绍如下。

1. 交互方式

交互方式即首先在操作系统提示符下输入 EXP，然后 EXPORT 工具会一步一步根据系统的提示输入导出参数(如用户名、口令和导出类型)，然后根据用户的回答，EXPORT 工具卸出相应的内容。

例题 14.4　采用交互方式将 system 用户下的学生表 student 和课程表 course 导出，导出的文件存放在 e:\orcl\student_course.dmp 中。

例题解析：

(1) 在命令提示符下输入 EXP，然后回车。

```
c:\> EXP
```

(2) 输入用户名和口令。

```
system/密码
```

(3) 输入数组读取缓冲区大小：

```
4096 >
```

这里使用默认值，直接回车即可。

(4) 输入导出文件名称。

```
EXPDAT.DMP > e:\orcl\student_course.dmp
```

(5) 选择要导出的类型，这里选择表 T。

```
E (整个数据库) (2)U(用户),或 (3)T(表): (2)U > T
```

(6) 导出权限(yes/no)：

```
yes >
```

使用默认值，选择 yes。

(7) 导出表数据(yes/no)：

```
yes >
```

使用默认设置，导出表数据。

(8) 压缩范围(yes/no)：

```
yes >
```

使用默认设置,压缩区。

(9) 要导出的表(T)或分区(T:P):

```
(按回车键退出)> student
在此输入要导出的表名称 student
正在导出表      student 导出了    10 行
```

(10) 要导出的表(T)或分区(T:P):

```
(按回车键退出)> course
在此输入要导出的表名称 course
正在导出表            student 导出了     8 行
…
```

继续导出其他表。

(11) 成功终止导出,没有出现警告。

2. 命令行方式

命令行就是将交互方式中所有用户回答的内容全部写在命令行上,每一个回答的内容作为某一关键字的值。

例题 14.5 采用命令行方式将 system 用户的学生表 student、课程表 course 和选课表 sc 导出到文件 e:\orcl\stu_cou_sc.dmp 中。

例题解析:

```
c:\ EXP USERID = system/密码
TABLES = (student,course,sc)
FILE = e:\orcl\stu_cou_sc.dmp;
```

3. 参数文件方式

参数文件就是存放上述关键字和相应值的一个文件,然后将该文件名作为命令行的 PARFILE 关键字的值。如果在参数文件中没有列出的关键字,该关键字就采用其默认值。

例题 14.6 采用参数文件方式将 system 用户的学生表 student 和课程表 course 两张表导出到文件 e:\orcl\stu_cou.dmp 中。

例题解析:

先用文本编辑器编辑一个参数文件,名为 c:\stu.TXT。

```
USERID = system/密码
TABLES = (student,course)
FILE = e:\orcl\stu_cou.dmp
```

执行下列命令完成备份操作。

```
c:\EXP  PARFILE = c:\stu.TXT;
```

14.4　Oracle 数据库的恢复

Oracle 数据库的恢复方法分为物理恢复与逻辑恢复。

14.4.1　物理恢复

物理恢复是针对物理文件的恢复。物理恢复又可分为数据库运行在非归档方式下的脱机物理恢复和数据库运行在归档方式下的联机物理恢复。

1. 非归档方式下的脱机恢复

一旦组成数据库的物理文件中有一个文件遭到破坏,必须在数据库关闭的情况下将全部物理文件装入到对应的位置上,进行恢复。

数据库的恢复一般分为 NOARCHIVELOG 模式和 ARCHIVELOG 模式,实际情况中很少会丢失整个 Oracle 数据库,通常只是一个驱动器损坏,仅仅丢失该驱动器上的文件。如何从这样的损失中恢复,很大程度上取决于数据库是否正运行在 ARCHIVELOG 模式下。如果没有运行在 ARCHIVELOG 模式下而丢失了一个数据库文件,就只能从最近的一次备份中恢复整个数据库,备份之后的所有变化都丢失,而且在数据库被恢复时,必须关闭数据库。由于在一个产品中丢失数据或者将数据库关闭一段时间是不可取的,所以大多数 Oracle 产品数据库都运行在 ARCHIVELOG 模式下。

2. 归档方式下的联机恢复

一旦这些数据文件中某一个遭到破坏,将该数据文件的备份装入到对应位置上,然后利用上次备份后产生的归档日志文件和联机日志文件进行恢复,可以恢复到失败这一刻。

具体实现步骤为:首先打开数据库,并确认数据库运行于归档模式,然后对数据库进行操作,接着将刚操作的内容归档到归档文件。此时如果组成数据库的物理文件中某一个数据文件遭到破坏,造成数据库无法启动,需要将被破坏的数据文件以前的备份按原路径装入。启动数据库到 MOUNT 状态,发 RECOVER 命令,系统自动利用备份后产生的归档日志文件进行恢复,恢复到所有数据文件序列号一致时为止。最后将此数据文件设为 ONLINE,并打开数据库到 OPEN 状态。

例题 14.7　将名为 orcl 的数据库进行归档模式的联机恢复(备份的文件已经存放在 e:\orcl\hot\目录下)。

例题解析:

(1) 启动数据库并确认数据库运行在自动归档模式。

```
SQL> CONNECT  / AS SYSDBA;
SQL> STARTUP;                        /* 启动数据库并保证运行于归档模式 */
SQL> ARCHIVE LOG LIST;               /* 验证数据库运行于归档模式 */
```

(2) 建立新用户 TEST 并授权,在 TEST 用户中建立 TEST 表,并往表中插入数据和

提交。

```
SQL> CREATE USER test                    /* 建立新用户 */
      IDENTIFIED BY test
      DEFAULT TABLESPACE users
      TEMPORARY TABLESPACE temp;
SQL> GRANT CONNECT,RESOURCE TO test;     /* 给用户授权 */
SQL> CONNECT test/test;                  /* 新用户连接 */
SQL> CREATE TABLE teacher                /* 建表 */
     (tno   NUMBER,
      tname   CHAR(8));
SQL> INSERT INTO teacher VALUES(1,'张三');/* 向表中插入数据 */
SQL> INSERT INTO teacher VALUES(2,'李四');
SQL> INSERT INTO teacher VALUES(3,'王五');
SQL> COMMIT;
SQL> DISCONNECT;
```

(3) 以sysdba权限登录,进行日志切换,以便将刚才所做的操作归档到归档日志文件。假设数据库有三个联机日志文件组,日志切换三次,保证刚插入的数据已被归档到归档日志文件。

```
SQL> CONNECT   / AS SYSDBA;
SQL> ALTER system SWITCH logfile;
SQL> ALTER system SWITCH logfile;
SQL> ALTER system SWITCH logfile;
```

(4) 关闭数据库,删除数据文件users01.dbf。

```
SQL> CONNECT   / AS SYSDBA;
SQL> SHUTDOWN;
SQL> HOST   DEL   D:\app\Administrator\oradata\orcl\USERS01.DBF;
```

(5) 执行打开数据库命令,发现错误,观察现象。

```
SQL> CONNECT   / AS SYSDBA;
SQL> STARTUP;
```

(6) 将归档模式下物理备份的USERS01.DBF文件装入到对应的目录。

```
SQL> HOST COPY e:\orcl\hot\USERS01.DBF   D:\app\Administrator\oradata\orcl\;
```

(7) 执行数据库恢复。

```
SQL> RECOVER   DATABASE   QUTO;
```

(8) 将USERS01.DBF文件置为online状态,以便执行下一步的查询操作。然后将数据库打开。

```
SQL> ALTER DATABASE datafile
  'D:\app\Administrator\oradata\orcl\USERS01.DBF' ONLINE;
SQL> ALTER DATABASE open;
```

（9）测试恢复后刚建立的表和插入的数据是否存在。说明数据库运行于归档模式时可以恢复到失败点之前。

```
SQL> CONNECT test/test;
SQL> SELECT * FROM teacher;
```

14.4.2　逻辑恢复

逻辑恢复是用 Oracle 系统提供的 IMPORT 工具将 EXPORT 工具存储在一个专门的操作系统文件中的内容按逻辑单元（表、用户、表空间、数据库）进行恢复。IMPORT 工具和 EXPORT 工具必须配套使用。根据卸出的四种模式（整个数据库模式、用户模式、表模式、表空间模式）可以分别装入整个数据库对象、装入某一用户的对象或者装入某一张表上的对象、表空间上的对象。装入运行方式有三种：交互式、命令行方式和参数文件方式。

例题 14.8　采用交互方式进行 system 用户下学生表 student 的导入（备份表已经存放在 e:\orcl\student_course.dmp 中）。

例题解析：

（1）在命令提示符下输入 IMP，然后回车。

```
c:\> IMP
```

（2）输入用户名和口令。

```
system/密码
```

（3）仅导入数据（yes/no）：

```
no >
```

这里使用默认值，直接回车即可。

（4）导入文件：

```
EXPDAT.DMP > e:\orcl\student_course.dmp
```

（5）输入插入缓冲区大小（最小为 8192 ）：

```
30720 >
```

这里使用默认值，直接回车即可。

（6）只列出导入文件的内容（yes/no）：

```
no >
```

这里使用默认值，直接回车即可。

(7) 由于对象已存在，忽略创建错误(yes/no)：

```
no > yes
```

(8) 导入权限(yes/no)：

```
yes >
```

这里使用默认值，直接回车即可。

(9) 导入表数据(yes/no)：

```
yes >
```

这里使用默认值，直接回车即可。

(10) 导入整个导出文件(yes/no)：

```
no >
```

这里使用默认值，直接回车即可。

(11) 用户名：

```
system
```

(12) 输入表(T)或分区(T:P)名称。空列表表示用户的所有表。

```
>   student
在此输入要导入的表名称 student
正在导入表      student 导入了   10 行
…
```

例题 14.9 采用命令行方式导入 e:\orcl\stu_cou_sc.dmp 文件中的学生表 student、课程表 course 和选课表 sc。

例题解析：

```
c:\IMP USERID = system/密码
TABLES = (student,course,sc)
ROWS = Y
FILE = e:\orcl\stu_cou_sc.dmp
```

例题 14.10 采用参数文件方式导入 e:\orcl\stu_cou.dmp 文件中的学生表 student 和课程表 course。

例题解析：

(1) 先用文本编辑器编辑一个参数文件，名为 c:\cou.TXT。

```
USERID = system/密码
```

```
TABLES = (student,course)
ROWS = Y
FILE = e:\orcl\stu_cou.dmp
```

（2）执行下列命令完成恢复操作。

```
c:\IMP  PARFILE = c:\cou.TXT;
```

14.5　实　　验

14.5.1　实验 1　数据库的备份

1. 实验目的

（1）理解数据库备份工作的重要性。

（2）了解数据库的备份原理。

（3）掌握常用的数据库备份技术。

2. 实验内容

（1）采用交互方式，使用 EXP 逻辑备份。

对 scott 用户进行逻辑备份。

```
c:\> EXP
…
```

（2）使用命令行方式备份。

以命令行方式将 scott 用户的 dept 和 emp 两张表导出到文件 EMP.DMP 中：

```
c:\EXP USERID = scott/tiger(密码)
TABLES = (dept,emp)
FILE = e:\orcl\emp.dmp
```

（3）使用参数文件方式备份。

用参数文件方式将 scott 用户的 dept 和 emp 两张表导出到文件 e:\orcl\f1.dmp 中。先用文本编辑器编辑一个参数文件，名为 c:\1.TXT，内容如下：

```
USERID = scott/tiger(密码)
TABLES = (dept,emp)
FILE = e:\orcl\f1.dmp
LOG = e:\orcl\log1.log
QUERY = "WHERE deptno IN (10,20,30)"
```

执行命令如下：

```
c:\EXP  PARFILE = c:\1.TXT
```

3. 考核标准

本实验为选做实验，根据课时进度安排，既可以在课堂上完成，也可作为学生课外作业独立完成。根据需求利用逻辑备份工具进行三种不同方式的备份，备份完全成功为优秀，如果出现错误，则根据错误情况灵活给分。

14.5.2 实验 2 数据库的恢复

1. 实验目的

(1) 理解数据库恢复工作的重要性。

(2) 了解数据库的恢复原理。

(3) 掌握常用的数据库恢复技术。

2. 实验内容

(1) 采用交互方式，使用 IMP 逻辑恢复。

对 scott 用户进行逻辑备份。

```
E:\> IMP
…
```

(2) 使用命令行方式恢复。

用命令行方式导入 e:\orcl\emp.dmp 文件中的 dept 和 emp 表。

```
C:\IMP  USERID = scott/tiger(密码)
TABLES = (dept,emp)
ROWS = Y
FILE = e:\orcl\emp.dmp
LOG = exp.log
```

(3) 使用参数文件方式恢复。

用参数文件方式导入 scott 用户的 dept 和 emp 表。

先编辑一个参数文件，名为 c:\2.TXT，内容如下：

```
USERID = system/(密码)
TABLES = (dept,emp)
ROWS = Y
FILE = e:\orcl\f1.dmp
LOG = e:\orcl\log1.log
FROMUSER = scott
TOUSER = system
```

然后执行 IMPORT 工具时使用该参数文件 c:\IMP PARFILE=c:\2.TXT。

3. 考核标准

本实验为选做实验，根据课时进度安排，既可以在课堂上完成，也可作为学生课外作业

独立完成。根据需求利用逻辑备份工具进行三种不同方式的恢复，恢复完全成功为优秀，如果出现错误，则根据错误情况灵活给分。

14.6　本章小结

本章首先介绍了事务的定义，事务的四个特性：原子性、一致性、隔离性和持续性，事务相关的控制语句。在 Oracle 中，没有提供开始事务处理语句，所有的事务都是隐式开始的，但是当用户想要终止一个事务处理时，必须显式使用 COMMIT 和 ROLLBACK 语句结束。

其次介绍了故障的种类，包括事务的内部故障、系统故障、介质故障和计算机病毒；恢复的实现技术，包括数据转储和日志文件。数据库恢复的策略与方法，包括事务故障的恢复、系统故障的恢复、介质故障的恢复和建立数据库镜像。

最后介绍了 Oracle 数据库物理备份中的冷备份和热备份方法、Oracle 数据库逻辑备份方法、Oracle 数据库脱机物理恢复和联机物理恢复方法和 Oracle 数据库的逻辑恢复方法。

14.7　课后习题

一、选择题

1. 事务是数据库的逻辑工作单位，事务中包括的各操作要么都做，要么都不做，这个特性为事务的(　　)。

A. 原子性　　B. 一致性　　C. 隔离性　　D. 持久性

2. SQL 语言中用(　　)语句实现事务的回滚。

A. CREATE TABLE　　B. ROLLBACK

C. GRANT　　D. COMMIT

3. 若系统在运行过程中，由于某种硬件故障，使存储在外存上的数据部分损失或全部损失，这种情况称为(　　)。

A. 事务故障　　B. 系统故障　　C. 介质故障　　D. 人为错误

4. 在 Oracle 数据库系统中，逻辑备份的命令为(　　)。

A. BACKUP　　B. LOG　　C. EXP　　D. IMP

5. 在 Oracle 数据库系统中，逻辑恢复的命令为(　　)。

A. BACKUP　　B. LOG　　C. EXP　　D. IMP

二、应用题

在 Oracle 中使用逻辑备份与恢复命令进行数据库中表的备份与恢复。

(1) 采用命令行方式将 system 用户的课程表 course 导出到文件 e:\orcl\course.dmp 中。

(2) 采用命令行方式导入 e:\orcl\course.dmp 文件中的课程表 course。

附录 A

样本数据库

本书中所涉及的所有案例均来自学生-课程数据库、员工-部门数据库。

1. 学生-课程数据库

该数据库包含学生表 student、课程表 course 和选课表 sc 三张表。

(1) 各表的结构如表 A.1～表 A.3 所示。

表 A.1 student(学生表)

字段名	字 段 类 型	是 否 为 空	说 明	字 段 描 述
SNO	CHAR(8)	NOT NULL	主键	学生学号
SNAME	VARCHAR2(20)		唯一	学生姓名
SEX	CHAR(4)	NOT NULL	非空	性别
AGE	INT		年龄大于 16 岁	年龄
DEPT	VARCHAR2(20)			学生所在的系别名称

表 A.2 course(课程表)

字段名	字 段 类 型	是 否 为 空	说 明	字 段 描 述
CNO	CHAR(8)	NOT NULL	主键	课程编号
CNAME	VARCHAR2(20)	NOT NULL	非空	课程名称
TNAME	VARCHAR2(20)			授课教师名
CPNO	CHAR(8)		外键(参照课程表中的课程编号)	先修课程号
CREDIT	NUMBER			学分

表 A.3 sc(选课表)

字段名	字 段 类 型	是 否 为 空	说 明	字 段 描 述
SNO	CHAR(8)	NOT NULL	外键(参照学生表中的学生编号)	学生学号
CNO	CHAR(8)	NOT NULL	外键(参照课程表中的课程编号)	课程编号
GRADE	NUMBER			选修成绩

其中,(Sno,Cno)属性组合为主键。

(2) 各表中的数据如图 A.1～图 A.3 所示。

SNO	SNAME	SEX	AGE	DEPT
11432001	陈一	男	17	侦查系
11432002	姚二	女	20	侦查系
11432003	张三	女	19	侦查系
11432004	李四	男	22	刑事技术系
11432005	王五	男	22	刑事技术系
11432006	赵六	男	19	刑事技术系
11432007	陈七	女	23	公安信息系
11432008	刘八	男	21	公安信息系
11432009	张九	女	18	治安管理系
11432010	孙十	女	21	治安管理系

图 A.1 学生表 student 中的数据

CNO	CNAME	TNAME	CPNO	CREDIT
c1	大学英语	李老师		3
c2	马克思主义基本原理	赵老师		2
c3	高等数学	曹老师		3
c4	证据学	杨老师		2
c5	罪犯心理矫治	张老师		2
c6	JAVA语言程序设计	唐老师	c3	3
c7	JSP程序设计	唐老师	c6	2
c8	公共安全危机管理	刘老师		2

图 A.2 课程表 course 中的数据

SNO	CNO	GRADE
11432001	c1	75
11432001	c2	95
11432002	c1	82
11432002	c2	88
11432002	c4	76
11432002	c5	55
11432003	c1	65
11432003	c2	72
11432003	c3	90
11432003	c4	85
11432004	c1	93
11432004	c2	96
11432004	c5	91
11432005	c1	50
11432005	c2	70
11432005	c5	88
11432005	c8	43
11432007	c1	75
11432007	c2	73
11432007	c3	66
11432007	c6	82
11432007	c7	94
11432008	c1	82
11432008	c2	77
11432008	c3	85
11432008	c4	87
11432008	c5	82
11432008	c6	94
11432008	c7	92
11432008	c8	89
11432009	c1	86
11432009	c2	
11432009	c8	
11432010	c1	73
11432010	c2	
11432010	c8	76

图 A.3 选课表 sc 中的数据

2. 员工-部门数据库

该数据库包含员工表 emp 和部门表 dept 两张表。

(1) 各表的结构如表 A.4、表 A.5 所示。

表 A.4 emp(员工表)

字段名	字段类型	是否为空	说　明	字段描述
EMPNO	CHAR(8)	NOT NULL	主键	员工编号
ENAME	VARCHAR2(20)			员工姓名
SEX	CHAR(4)			性别
AGE	NUMBER			年龄
JOB	VARCHAR2(20)			职位
MGR	CHAR(8)		外键(参照员工表中的员工编号)	主管经理编号
SALARY	NUMBER			月薪
DEPTNO	CHAR(8)		外键(参照部门表中的部门编号)	部门编号

表 A.5 dept(部门表)

字段名	字段类型	是否为空	说　明	字段描述
DEPTNO	CHAR(8)	NOT NULL	主键	部门编号
DNAME	VARCHAR2(20)		唯一	部门名称
LOC	VARCHAR2(20)			部门所在地点

(2) 各表中的数据如图 A.4、图 A.5 所示。

EMPNO	ENAME	SEX	AGE	JOB	MGR	SALARY	DEPTNO
1001	陈一	男	52	总经理		9500	
2001	姚二	男	47	部门经理	1001	6000	10
2002	张三	男	35	会计	2001	4500	10
2003	李四	女	27	出纳	2001	3500	10
3001	王五	女	38	部门经理	1001	5500	20
3002	赵六	女	27	文员	3001	2500	20
3003	陈七	女	23	文员	3001	1800	20
4001	刘八	男	40	部门经理	1001	6500	30
4002	张九	男	35	业务员	4001	3500	30
4003	孙十	女	24	业务员	4001	2400	30
5001	沈一	男	38	部门经理	1001	6500	40
5002	吴二	女	32	程序员	5001	3500	40
5003	崔三	男	27	程序员	5001	3000	40
5004	刘四	男	19	程序员	5001	1800	40
6001	张五	女	35	部门经理	1001	4000	50
6002	韩六	男	26	维修员	6001	2200	50

图 A.4 员工表 emp 中的数据

DEPTNO	DNAME	LOC
10	财务部	上海
20	人力资源部	北京
30	销售部	北京
40	研发部	上海
50	客服部	大连

图 A.5 部门表 dept 中的数据

附录 B

Oracle 11g 数据库的安装和卸载

1. 安装 Oracle 11g 数据库

(1) 在 Oracle 11g 的安装程序文件夹中找到可执行安装文件 setup. exe，并双击安装，如图 B. 1 和图 B. 2 所示。



图 B. 1　Oracle 11g 安装程序文件夹



图 B. 2　Oracle 11g 安装初始化

(2) 配置安全更新，可以填写自己的电子邮件地址，用于接收一些邮件信息。取消选中“我希望通过 My Oracle Support 接收安全更新(W)”复选框，如图 B. 3 和图 B. 4 所示。

(3) 安装选项，选择创建和配置数据库，如图 B. 5 所示。在数据库管理软件安装完毕后，系统会自动创建一个数据库实例。



图 B.3　初始的配置安全更新页面



图 B.4　设置后的配置安全更新页面



图 B.5　选择安装选项

(4) 系统类,选择默认的桌面类,如图 B.6 所示。



图 B.6　选择系统类

(5) 典型安装配置,如果修改目录路径,则注意路径中不要含有中文或其他特殊字符。全局数据库名默认为 orcl,管理口令需要自己设置,并且要牢牢记住。口令密码在设定时,会有提示警告,要求口令密码必须由大写字母、小写字母和数字混合组成,并且必须是 8 位以上,如图 B.7 和图 B.8 所示。



图 B.7 初始的典型安装配置页面

(6) 先决条件检查,安装程序会检查软硬件系统是否满足安装此 Oracle 版本的最低要求。在检测过程中,如果出现部分失败,则需要用户手动验证,可以单击"全部忽略"按钮,再单击"下一步"按钮,才可以继续进行安装,如图 B.9 所示。

(7) 概要信息,即安装前的一些相关配置信息展示,如图 B.10 所示。

(8) 安装产品界面,这个阶段会自动进行安装,耗时较长,如图 B.11 所示。

(9) 数据库管理软件安装完毕后,会自动创建一个名称为 orcl 的数据库实例,如图 B.12 所示。

(10) 当 Oracle 配置助手运行完后,系统会弹出一个关于已生成数据库的信息,如图 B.13 所示。

(11) 安装完成后,将打开"安装成功"窗口,如图 B.14 所示。单击"关闭"按钮,结束安装过程。

图 B.8　修改后的典型安装配置页面



图 B.9　先决条件检查



图 B.10 概要信息



图 B.11 安装数据库管理软件



图 B.12　创建数据库实例

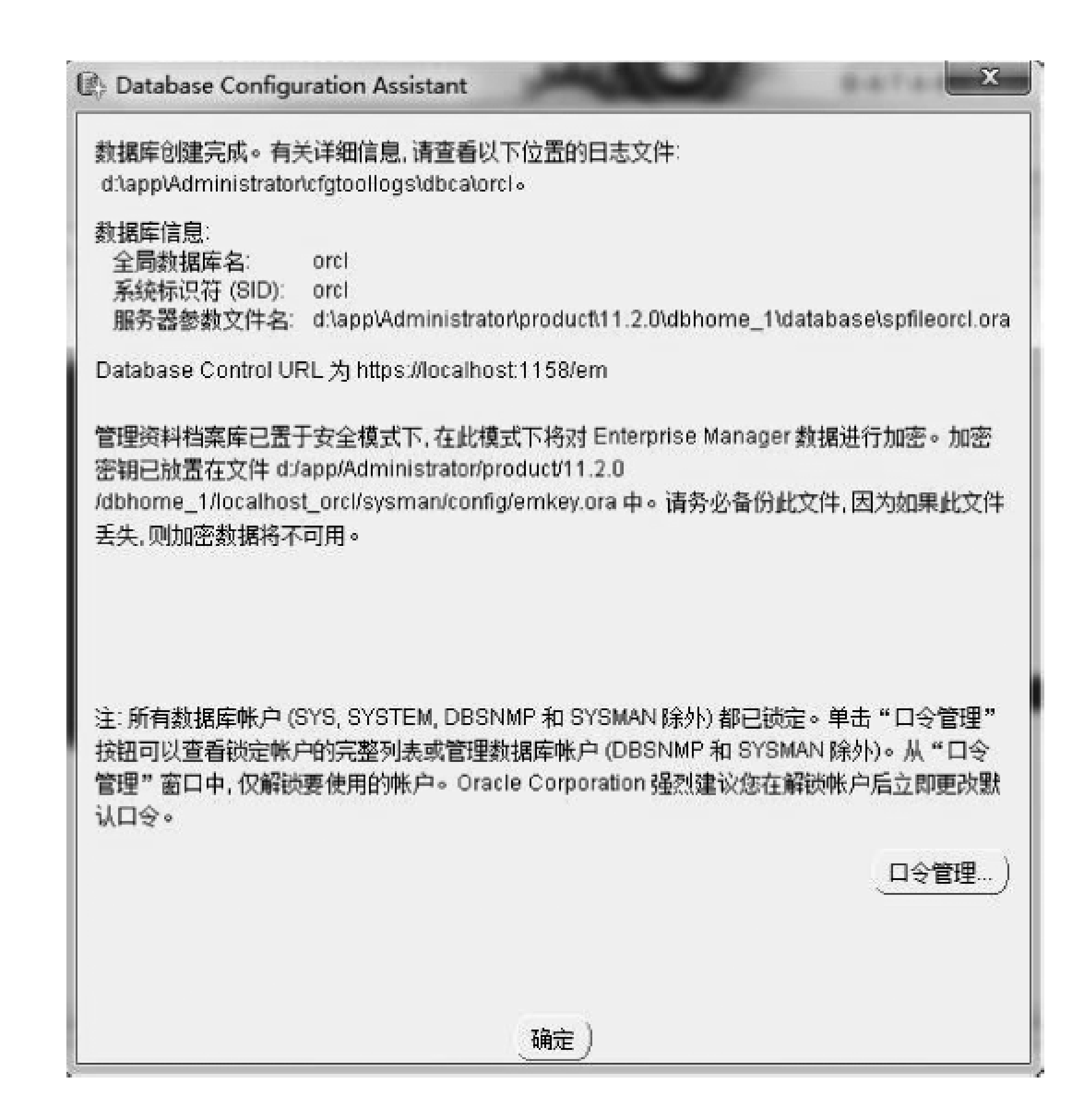


图 B.13　数据库配置信息



图 B.14　数据库安装结束

2. 查看安装情况

(1) Oracle 11g 安装后的目录结构,在数据库实例 oradata\\orcl 文件夹中存储物理文件,包括数据文件.dbf、控制文件.ctl、重做日志文件.log,如图 B.15 所示。



图 B.15　Oracle11g 安装后的目录结构

(2) 查看"服务"管理器中相关的 Oracle 服务。为了提高系统的性能,可以将 Oracle 服务设置为手动启动,根据自己的需求启动相应服务即可,如图 B.16 所示。

(3) 查看注册表,具体目录结构如图 B.17 所示。

(4) 在"开始"→"所有程序"中增加 Oracle-OracleDb11g_home1 文件夹,结果如图 B.18 所示。

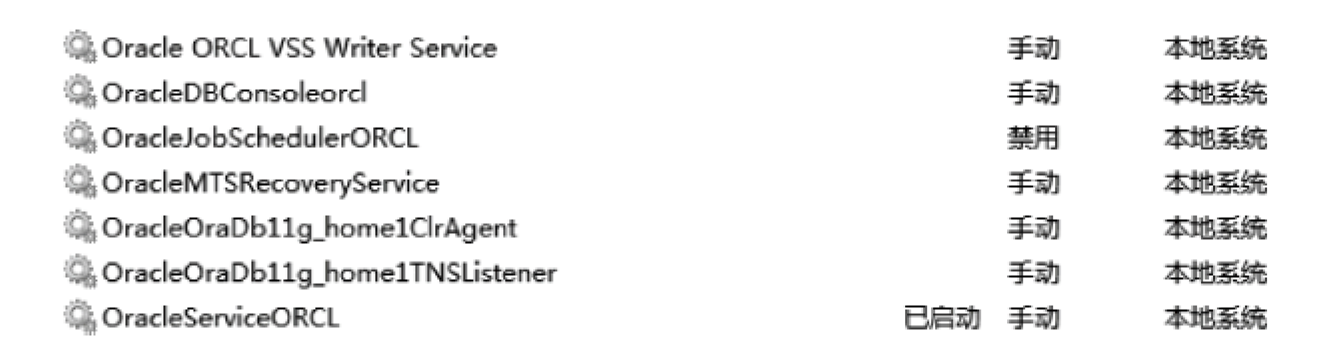


图 B.16　Oracle 服务



图 B.17　注册表　　　图 B.18　开始程序

3. 卸载 Oracle 11g 数据库

(1) 在“服务”窗口中，停止所有以 Oracle 开头的服务。

(2) 单击“开始”→“所有程序”→Oracle-OracleDb11g_home1\“Oracle 安装产品”→Universal Installer 命令，打开 Oracle Universal Installer(OUI)窗口，如图 B.19 和图 B.20 所示。

(3) 单击“卸装产品”按钮，不要单击“下一步”按钮，打开“产品清单”窗口，如图 B.21 所示。

(4) 在产品清单窗口中，单击全部展开，除了 OraDb11g_home1 外，选中其他项目，如图 B.22 所示。

(5) 选中要删除的 Oracle 产品后，单击“删除”按钮，打开确认删除窗口，如图 B.23 所示。

(6) 单击“是”按钮，开始删除 Oracle 产品，如图 B.24 所示。

(7) 卸载完毕后，Oracle 在产品清单中消失，如图 B.25 所示。

(8) 运行 regedit 命令，打开注册表窗口，如图 B.26 所示。删除注册表中与 Oracle 相关的内容。

① 依次展开 HKEY_LOCAL_MACHINE\SOFTWARE，删除 oracle 目录。

② 依次展开 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services，删除所有 oracle 开头的目录。

③ 依次展开 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\Application，删除所有 oracle 开头的目录。

④ 依次展开 HKEY_CLASSES_ROOT，删除所有以 ora 开头的目录。

(9) 删除“开始”→“所有程序”菜单中所有 Oracle 的组和图标。

(10) 重新启动计算机，删除所有与 Oracle 相关的目录。把 Oracle 的安装目录 app 和操作系统目录(Program Files)下的 Oracle 目录一并删除。至此，Oracle11g 数据库完全卸载完毕。



图 B.19 条件检查



图 B.20 Oracle Universal Installer(OUI)窗口



图 B.21 产品清单



图 B. 22　选择要删除的 Oracle 产品



图 B. 23　"确认"界面

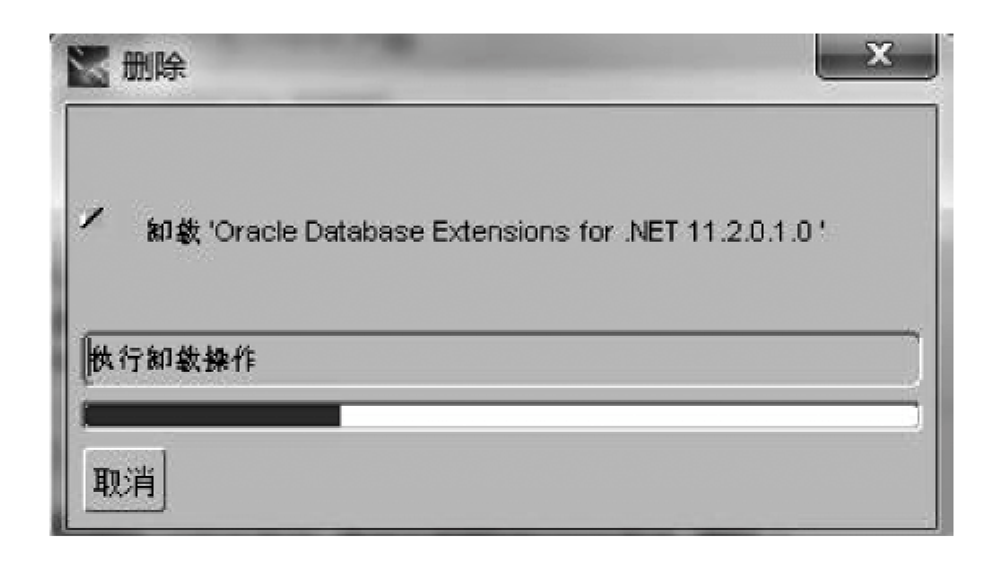


图 B. 24　"删除"界面



图 B.25　卸载后的产品清单



图 B.26　“注册表编辑器”界面

附录 C

实验参考答案

第 3 章实验

实验 1　SQL * PLUS 常用命令练习

(1) SHOW USER;

(2) SELECT table_name FROM user_tables;

(3) DESC emp;

(4) HELP INDEX;

(5) ? RUN;

(6) SET LINESIZE 200;
SET PAGESIZE 200;

(7) list;

(8) / , run ,r;

(9) CHANGE/FOM/FROM;

(10) EDIT;

(11) SAVE c:\part1; (默认保存成.sql)
SAVE c:\part1.txt;

(12) GET c:\part1.sql;

(13) START c:\part1.sql;

(14) SPOOL c:\part2.sql;(先创建文本,从想保存的位置开始)
SELECT * FROM emp;(写入想保存的命令,包括结果)
SPOOL OFF;(操作结束的位置)

实验 2　数据定义语言 DDL

(1) 按要求采用不同的约束类型创建科室表和医生表。

①
```
CREATE TABLE dept
(deptno CHAR(10) PRIMARY KEY,
dname VARCHAR(15) UNIQUE,
```

```
loc VARCHAR(20));
```

②
```
CREATE TABLE doctor
(docno CHAR(10) PRIMARY KEY,
docname VARCHAR(15) NOT NULL,
age INT CHECK(age BETWEEN 18 AND 60),
sal NUMBER,
deptno CHAR(10) REFERENCES dept(deptno));
```

(2) 按要求对于表的结构进行修改。

① `ALTER TABLE doctor ADD birthday DATE;`

② `ALTER TABLE dept MODIFY dname VARCHAR2(20);`

③ `ALTER TABLE doctor ADD CONSTRAINT CHK_SAL CHECK(sal BETWEEN 1000 AND 8000);`

④ `ALTER TABLE doctor DROP COLUMN birthday;`

(3) 按要求删除基本表。

```
DROP TABLE doctor;
```

实验3 数据操纵语言DML

(1) 创建教师信息基本表。

```
CREATE TABLE teacher
(tno CHAR(8) PRIMARY KEY,
tname VARCHAR2(20) NOT NULL,
tsex VARCHAR2(6),
tsal NUMBER CHECK(tsal > 1800),
tdept CHAR(20));
```

(2) 练习向基本表中插入数据、修改数据和删除数据。

①
```
INSERT INTO teacher VALUES('T001','张老师','女',3000,'侦查系');
INSERT INTO teacher VALUES('T002','王老师','男',2800,'侦查系');
INSERT INTO teacher VALUES('T003','李老师',NULL,NULL,'公安信息系');
INSERT INTO teacher VALUES('T004','张老师','男',3500,'公安信息系');
INSERT INTO teacher VALUES('T005','刘老师','女',2200,'治安管理系');
```

② `UPDATE teacher SET tdept = '信息系' WHERE tdept = '公安信息系';`

③ `UPDATE teacher SET tsal = 3300 WHERE tname = '王老师';`

④ `DELETE FROM teacher WHERE tdept = '侦查系';`

⑤ `DELETE FROM teacher;`

实验4 单表查询

(1) 创建员工信息表。

```
CREATE TABLE employees
```

```
(eno CHAR(8) PRIMARY KEY,
ename VARCHAR2(10) NOT NULL,
sex CHAR(6) CHECK(sex = '男' or sex = '女'),
age INT CHECK (age > 18),
job VARCHAR2(20),
sal NUMBER,
dept VARCHAR2(20));
```

(2) 向已创建的员工表中插入数据。

```
INSERT INTO employees VALUES('1001','张三','男',20,'销售',1000,'市场部');
INSERT INTO employees VALUES('1002','李四','女',26,'会计',1600,'财务部');
INSERT INTO employees VALUES('1003','王五','女',22,'销售',1000,'市场部');
INSERT INTO employees VALUES('1004','赵六','男',19,NULL,NULL,NULL);
INSERT INTO employees VALUES('1005','张七','女',23,'测试',1400,'技术部');
INSERT INTO employees VALUES('1006','赵八','男',30,'研发',2000,'技术部');
```

(3) 按要求完成各种单表信息查询,并验证聚集函数的功能。

① SELECT ename,sex,sal FROM employees;

② SELECT DISTINCT dept FROM employees;

③ SELECT ename,2015 - age FROM employees WHERE dept = '技术部';

④ SELECT ename,age FROM employees WHERE sal > 1200;

⑤ SELECT ename,sal FROM employees WHERE age NOT BETWEEN 20 AND 25;

⑥ SELECT ename,sex FROM employees WHERE dept IN ('财务部','技术部');

⑦ SELECT ename,age,job FROM employees WHERE ename LIKE '张%';

⑧ SELECT ename,age FROM employees WHERE job IS NULL;

⑨ SELECT ename FROM employees WHERE dept = '市场部' AND age < 25 AND sex = '男';

⑩ SELECT ename,sal FROM employees WHERE age > 20 ORDER BY sal DESC;

⑪ SELECT COUNT(*) FROM employees WHERE dept = '市场部';

⑫ SELECT MAX(sal) FROM employees;

⑬ SELECT MIN(sal) FROM employees;

⑭ SELECT AVG(age) FROM employees WHERE dept = '技术部';

⑮ SELECT SUM(sal) FROM employees WHERE dept = '市场部';

实验5 多表连接查询和集合查询

(1) 根据样本数据库中的表和数据,进行多表连接查询操作的练习。

① SELECT sname,grade FROM student,sc WHERE student.sno = sc.sno AND cno = 'c4' ORDER BY grade DESC;

② SELECT COUNT(*),AVG(age) FROM student WHERE SEX = '男';

③ SELECT sno,COUNT(cno),AVG(grade) FROM sc GROUP BY sno;

④ SELECT sno,AVG(grade) FROM sc GROUP BY sno HAVING AVG(grade)>80;

⑤ SELECT sno,grade FROM course,sc WHERE course.cno=sc.cno AND cname='JSP 程序设计';

⑥ SELECT sname,cname,gradeFROM student,sc,course WHERE student.sno=sc.sno AND course.cno=sc.cno AND dept='侦查系';

(2) 根据样本数据库中的表和数据,进行集合查询操作的练习。

① SELECT * FROM student WHERE dept='侦查系'
UNION
SELECT * FROM student WHERE dept='刑事技术系';

② SELECT sno FROM student WHERE dept='公安信息系'
INTERSECT
SELECT sno FROM sc WHERE cno='c4';

③ SELECT * FROM student WHERE dept='治安管理系'
MINUS
SELECT * FROM student WHERE age<=20;

实验 6 嵌套查询

(1) 根据样本数据库中的表和数据,进行不相关子查询的练习。

① SELECT sname,age FROM STUDENT WHERE dept=(SELECT dept FROM student WHERE sname='王五');

② SELECT sno,sname FROM student WHERE sno IN(SELECT sno FROM sc WHERE cno IN(SELECT cno FROM course WHERE cname='公共安全危机管理'))

③ SELECT * FROM student WHERE age>ALL(SELECT age FROM student WHERE dept='侦查系');

④ SELECT sno FROM sc GROUP BY sno HAVING AVG(grade)=(SELECT MAX(AVG(grade)) FROM sc GROUP BY sno);

⑤ SELECT cno FROM course WHERE cno NOT IN(SELECT cno FROM sc WHERE sno IN (SELECT sno FROM student WHERE sname='李四'));

(2) 根据样本数据库中的表和数据,进行相关子查询的练习。

① SELECT sno,grade FROM SC WHERE EXISTS (SELECT * FROM course WHERE course.cno=sc.cno AND cname='证据学');

② SELECT sno,sname FROM student WHERE NOT EXISTS(SELECT * FROM sc WHERE student.sno=sc.sno AND cno='c2');

③ SELECT sname FROM student WHERE sno IN (SELECT sno FROM sc WHERE NOT EXISTS(SELECT * FROM sc WHERE student.sno=sc.sno AND grade<=70));

实验 7 视 图

(1) CREATE VIEW crim_student AS SELECT sno,sname,age FROM student WHERE dept='刑事技术系';

(2) SELECT sname FROM crim_student WHERE age>20;

(3) UPDATE crim_student SET sname='李子' WHERE sno='11432004';

(4) DELETE FROM crim_student WHERE sname = '赵六';

(5) DROP VIEW crim_student;

第6章实验

实验1 PL/SQL 基本结构

(1) 练习简单的变量定义与赋值。

```
DECLARE
    a NUMBER: = 2;
    b NUMBER: = 3;
    c NUMBER;
BEGIN
    c: = a + b;
    DBMS_OUTPUT.PUT_LINE('c的值为:'||c);
END;
```

(2) 编写简单的 PL/SQL 程序块,进行变量定义、变量输入,变量赋值与变量输出的练习。

①

```
DECLARE
    v_grade sc.grade%TYPE;
BEGIN
    SELECT AVG(grade) INTO v_grade FROM student,sc WHERE student.sno = sc.sno AND sname = '王五';
    DBMS_OUTPUT.PUT_LINE('王五同学的平均成绩为:'||v_grade);
END;
```

②

```
DECLARE
    v_sno student.sno%TYPE: = &sno;
    v_num NUMBER: = 0;
BEGIN
    SELECT COUNT(*) INTO v_num FROM sc WHERE sno = v_sno;
    DBMS_OUTPUT.PUT_LINE('此同学选修了:'||v_num||'门课程');
END;
```

实验2 PL/SQL 条件语句

IF 语句:

```
DECLARE
```

```
    v_sno student.sno%TYPE: = &sno;
    v_num NUMBER: = 0;
BEGIN
    SELECT COUNT( * ) INTO v_num FROM sc WHERE sno = v_sno;
    IF v_num > 6 THEN
        DBMS_OUTPUT.PUT_LINE('此同学的选课门数为:'||v_num||',你真是个爱学习的好孩子');
    ELSIF v_num < 3 THEN
        DBMS_OUTPUT.PUT_LINE('此同学的选课门数为:'||v_num||',你真是个小懒蛋');
    ELSE
        DBMS_OUTPUT.PUT_LINE('此同学的选课门数为:'||v_num||',加油吧');
    END IF;
END;
```

CASE 语句:

```
DECLARE
    v_sno student.sno%TYPE: = &sno;
    v_num NUMBER: = 0;
BEGIN
    SELECT COUNT( * ) INTO v_num FROM sc WHERE sno = v_sno;
    CASE
        WHEN v_num > 6 THEN
            DBMS_OUTPUT.PUT_LINE('此同学的选课门数为:'||v_num||',你真是个爱学习的好孩子');
        WHEN v_num < 3 THEN
            DBMS_OUTPUT.PUT_LINE('此同学的选课门数为:'||v_num||',你真是个小懒蛋');
        ELSE
            DBMS_OUTPUT.PUT_LINE('此同学的选课门数为:'||v_num||',加油吧');
    END CASE;
END;
```

实验 3 PL/SQL 循环语句

WHILE 循环:

```
DECLARE
    v_sum NUMBER: = 0;
    i NUMBER: = 0;
BEGIN
    WHILE i <= 20   LOOP
        v_sum: = v_sum + i;
        i: = i + 2;
    END LOOP;
DBMS_OUTPUT.PUT_LINE(v_sum);
END;
```

FOR 循环:

```
DECLARE
    v_sum NUMBER: = 0;
BEGIN
    FOR i in 1..20 LOOP
        IF mod(i,2) = 0 THEN
            v_sum: = v_sum + i;
        END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE (v_sum);
END;
```

第 7 章实验

实验 1　系统预定义异常

（1）

```
DECLARE
    v_cname course.cname%TYPE: = &p_cname;
    v_credit course.credit%TYPE;
BEGIN
    SELECT credit INTO v_credit FROM course WHERE cname = v_cname;
    DBMS_OUTPUT.PUT_LINE(v_cname||'课程的学分是：'||v_credit);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('查无此课!');
END;
```

（2）

```
DECLARE
    v_dept student.dept%TYPE;
BEGIN
    SELECT dept INTO v_dept FROM student WHERE sname = '杨珺婷';
    DBMS_OUTPUT.PUT_LINE('该同学的系别为：'||v_dept);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('捕获到了 NO_DATA_FOUND 异常');
        DBMS_OUTPUT.PUT_LINE('SELECT 语句未找到相应的记录!');
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('捕获到了 TOO_MANY_ROWS 异常');
        DBMS_OUTPUT.PUT_LINE('SELECT 语句检索到多行数据!');
END;
```

实验 2　用户自定义异常

(1)

```
DECLARE
    v_num NUMBER;
    e EXCEPTION;
BEGIN
    SELECT COUNT( * ) INTO v_num FROM student WHERE dept = '侦查系';
    IF v_num > 200 THEN
        DBMS_OUTPUT.PUT_LINE('it is big');
    ELSIF v_num > 100 THEN
        DBMS_OUTPUT.PUT_LINE('it is middle');
    ELSE
        DBMS_OUTPUT.PUT_LINE('it is small');
    END IF;
    IF v_num < 10 THEN
        RAISE e;
    END IF;
EXCEPTION
    WHEN e THEN
        DBMS_OUTPUT.PUT_LINE('此系应扩大招生');
END;
```

(2)

```
DECLARE
    v_sno sc.sno%TYPE: = &p_sno;
    v_count NUMBER;
    e1 EXCEPTION;
    e2 EXCEPTION;
BEGIN
    SELECT COUNT( * ) INTO v_count FROM sc WHERE sno = v_sno AND grade < 60;
    IF v_count > 3 THEN
        RAISE e1;
    ELSIF v_count >= 2 THEN
        RAISE e2;
    ELSE
        DBMS_OUTPUT.PUT_LINE('不及格课程数为: '||v_count);
    END IF;
EXCEPTION
    WHEN e1 THEN
        DBMS_OUTPUT.PUT_LINE('留级');
    WHEN e2 THEN
        DBMS_OUTPUT.PUT_LINE('跟班试读');
END;
```

第 8 章实验

实验 1 不带参数的游标

(1)

```
DECLARE
    v_dept student.dept%type:=&p_dept;
    v_sname student.sname%type;
    v_grade sc.grade%TYPE;
    CURSOR student_cursor  IS  SELECT  sname, grade FROM student, course, sc WHERE student.
sno=sc.sno AND course.cno=sc.cno AND dept=v_dept AND cname='大学英语';
BEGIN
    OPEN student_cursor;
    LOOP
        FETCH student_cursor INTO v_sname,v_grade;
        EXIT WHEN student_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('学生姓名为:'||v_sname||','||'成绩为:'||v_grade);
    END LOOP;
    CLOSE student_cursor;
END;
```

(2)

```
DECLARE
    v_dept student.dept%TYPE:=&p_dept;
    CURSOR student_cursor  IS  SELECT  sname, grade FROM student, course, sc WHERE student.
sno=sc.sno AND course.cno=sc.cno AND dept=v_dept AND cname='JSP 程序设计';
BEGIN
    FOR a IN student_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('学生姓名为:'||a.sname||','||'成绩为:'||a.grade);
    END LOOP;
END;
```

(3)

```
DECLARE
    CURSOR emp_cursor IS SELECT ename,salary FROM emp,dept WHERE emp.deptno=dept.deptno AND
dname='研发部' FOR UPDATE OF salary;
BEGIN
    FOR emp_record IN emp_cursor LOOP
    IF emp_record.salary<3000 THEN
        UPDATE emp SET salary=3000 WHERE CURRENT OF emp_cursor;
    END IF;
```

```
    DBMS_OUTPUT.PUT_LINE(emp_record.ename||emp_record.salary);
    END LOOP;
END;
```

实验 2 带参数的游标

(1)

```
DECLARE
    CURSOR stu_cursor (v_dept char) IS SELECT sno,sname FROM student WHERE dept = v_dept;
    stu_record stu_cursor%ROWTYPE;
BEGIN
    OPEN stu_cursor('治安管理系');
    LOOP
        FETCH  stu_cursor INTO stu_record;
        EXIT  WHEN stu_cursor%NOTfound ;
        DBMS_OUTPUT.PUT_LINE('学号为:'||stu_record.sno||','||'姓名为:'||stu_record.sname);
    END LOOP;
    CLOSE stu_cursor;
END;
```

(2)

```
DECLARE
    CURSOR stu_cursor IS SELECT * FROM student;
    CURSOR student_cursor (v_dept char) IS SELECT * FROM student WHERE  dept = v_dept ORDER BY
age DESC;
BEGIN
    FOR stu_record IN stu_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('名称系别为:'||stu_record.dept);
        FOR student_record IN student_cursor(stu_record.dept) LOOP
            DBMS_OUTPUT.PUT_LINE('学生姓名为:'||student_record.sname||','||'年龄为: '||
student_record.age);
        END LOOP;
    END LOOP;
END;
```

实验 3 隐 式 游 标

(1)

```
BEGIN
DELETE FROM sc WHERE sno = &p_sno;
    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('删除失败!');
    ELSE
```

```
        DBMS_OUTPUT.PUT_LINE('删除成功!');
        DBMS_OUTPUT.PUT_LINE('删除的行数为: '||SQL%ROWCOUNT);
    END IF;
END;
```

(2)

```
BEGIN
UPDATE course SET cname = 'Oracle',credit = 4 where cno = 'c9';
    IF SQL%NOTFOUND THEN
        INSERT INTO course(cno,cname,credit) VALUES('c9','Oracle',4);
        DBMS_OUTPUT.PUT_LINE('没有找到要更新的记录,插入新信息');
    END IF;
END;
```

第 9 章实验

实验 1　不带参数的存储过程

(1)

```
CREATE OR REPLACE PROCEDURE p_count
IS
    v_num NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_num FROM student;
    DBMS_OUTPUT.PUT_LINE('总人数为: '||v_num);
END p_count;
```

(2)

```
EXECUTE p_count;
```

实验 2　带参数的存储过程

(1) 练习带 in 型参数的存储过程的创建与调用。

①

```
CREATE OR REPLACE PROCEDURE p_stu
(v_sno IN student.sno%TYPE)
IS
    v_sname student.sname%TYPE;
    v_age student.age%TYPE;
BEGIN
```

```
    SELECT sname,age INTO v_sname,v_age FROM student WHERE sno = v_sno;
    DBMS_OUTPUT.PUT_LINE('姓名: '||v_sname||',年龄: '||v_age);
END p_stu;
```

②

```
BEGIN
     p_stu('11432001');
END;
```

(2) 练习带 in 型参数的存储过程的创建与调用。

①

```
CREATE OR REPLACE PROCEDURE p_dept
(v_dept IN student.dept%TYPE)
IS
    CURSOR student_cursor IS SELECT sno,sname FROM student WHERE dept = v_dept;
BEGIN
    FOR student_record IN student_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('学生学号:'||student_record.sno||','||'姓名: '||student_
record.sname);
    END LOOP;
END p_dept;
```

②

```
BEGIN
     p_dept('公安信息系');
END;
```

(3) 练习带 in 与 out 型参数的存储过程的创建与调用。

①

```
CREATE OR REPLACE PROCEDURE p_grade
(v_sno IN student.sno%TYPE,
 v_grade OUT NUMBER)
IS
BEGIN
    SELECT AVG(grade) INTO v_grade FROM sc WHERE sno = v_sno;
END p_grade;
```

②

```
DECLARE
v_grade NUMBER;
BEGIN
     p_grade('11432002',v_grade);
```

```
    DBMS_OUTPUT.PUT_LINE('平均成绩为:'|| v_grade);
END;
```

第 10 章实验

实验 1　不带参数的存储函数

（1）创建一个返回学生总人数的无参数函数。

```
CREATE OR REPLACE FUNCTION f_count
RETURN NUMBER
IS
    v_count NUMBER;
BEGIN
    SELECT COUNT( * ) INTO v_count FROM student;
    RETURN v_count;
END f_count;
```

（2）调用此存储函数：输出当前学生的总人数。

```
DECLARE
    v_num NUMBER;
BEGIN
    v_num: = f_count;
    DBMS_OUTPUT.PUT_LINE('学生总人数为:'||v_num);
END;
```

实验 2　带参数的存储函数

（1）练习带 in 型参数的存储函数的创建与调用。

①

```
CREATE OR REPLACE FUNCTION f_grade
(v_sno student.sno%TYPE)
RETURN number
IS
    v_num NUMBER;
BEGIN
    SELECT SUM(grade) INTO v_num FROM sc WHERE sno = v_sno;
    RETURN v_num;
END f_grade;
```

②

```
DECLARE
    v_sum NUMBER;
BEGIN
    v_sum: = f_grade('11432002');
    DBMS_OUTPUT.PUT_LINE('该学生的总成绩为:'||v_sum);
END;
```

(2) 练习带 in 型参数的存储函数的创建与调用。

①

```
CREATE OR REPLACE FUNCTION f_stu
(v_sno student.sno%TYPE)
RETURN student%ROWTYPE
IS
    stu_record student%ROWTYPE;
BEGIN
    SELECT * INTO stu_record FROM student WHERE sno = v_sno;
    RETURN stu_record;
END f_stu;
```

②

```
DECLARE
    stu student%ROWTYPE;
BEGIN
    stu: = f_stu('11432003');
    DBMS_OUTPUT.PUT_LINE('姓名为:'||stu.sname||'性别为:'||stu.sex||'年龄为:'||stu.age);
END;
```

第 11 章实验

实验 1　包的创建与调用

(1)

包说明的创建:

```
CREATE OR REPLACE PACKAGE emp_package
IS
    PROCEDURE get_mgr(v1_deptno IN emp.deptno%TYPE,mgr_ename OUT emp.ename%TYPE);
    FUNCTION get_count(v2_deptno emp.deptno%TYPE)
    RETURN NUMBER;
END emp_package;
```

包主体的创建：

```
CREATE OR REPLACE PACKAGE BODY emp_package
IS
    PROCEDURE get_mgr(v1_deptno IN emp.deptno%TYPE,mgr_ename OUT emp.ename%TYPE)
    IS
    BEGIN
        SELECT ename INTO mgr_ename FROM emp WHERE job='部门经理' AND deptno=v1_deptno;
    END;
    FUNCTION get_count(v2_deptno emp.deptno%TYPE)
    RETURN NUMBER
    IS
        v_count NUMBER;
    BEGIN
        SELECT COUNT(*) INTO v_count FROM emp WHERE deptno=v2_deptno;
    RETURN v_count;
    END;
END emp_package;
```

(2)

```
DECLARE
    s_ename emp.ename%TYPE;
BEGIN
    emp_package.get_mgr('30',s_ename);
    DBMS_OUTPUT.PUT_LINE('该部门的经理为:'||s_ename);
END;
```

(3)

```
DECLARE
    s_count NUMBER;
BEGIN
    s_count:=emp_package.get_count('30');
    DBMS_OUTPUT.PUT_LINE('该部门的员工人数为:'||s_count);
END;
```

实验 2　包的重载

(1)

包说明的创建：

```
CREATE OR REPLACE PACKAGE pkg_overload
```

```
AS
    PROCEDURE get_dept(v_deptno NUMBER);
    PROCEDURE get_dept(v_dname dept.dname%TYPE);
END pkg_overload;
```

包主体的创建：

```
CREATE OR REPLACE PACKAGE BODY pkg_overload
AS
    PROCEDURE get_dept(v_deptno NUMBER)
    AS
        v_dept dept%ROWTYPE;
    BEGIN
        SELECT * INTO v_dept FROM dept WHERE deptno = v_deptno;
        DBMS_OUTPUT.PUT_LINE(v_dept.dname||','||v_dept.loc);
    END get_dept;
    PROCEDURE get_dept(v_dname dept.dname%TYPE)
    AS
        v_dept dept%ROWTYPE;
    BEGIN
        SELECT * INTO v_dept FROM dept WHERE dname = v_dname;
        DBMS_OUTPUT.PUT_LINE(v_dept.deptno||','||v_dept.loc);
    END get_dept;
END pkg_overload;
```

(2)

```
BEGIN
    pkg_overload.get_dept(30);
END;
```

(3)

```
BEGIN
    pkg_overload.get_dept('财务部');
END;
```

第 12 章实验

实验 1　语句级触发器

(1)

```
CREATE OR REPLACE TRIGGER t1
AFTER UPDATE ON student
```

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('您执行了更新数据的操作…');
END t1;
```

（2）

```
CREATE OR REPLACE TRIGGER t2
AFTER INSERT OR UPDATE OR DELETE ON student
DECLARE
    v_count NUMBER;
    avg_age student.age%TYPE;
    max_age student.age%TYPE;
BEGIN
    IF INSERTING THEN
        SELECT COUNT(*) INTO v_count FROM student;
        DBMS_OUTPUT.PUT_LINE('插入后的学生人数为:'||v_count);
    END IF;
    IF UPDATING THEN
        SELECT AVG(age) INTO avg_age FROM student;
        DBMS_OUTPUT.PUT_LINE('学生平均年龄为:'||avg_age);
    END IF;
    IF DELETING THEN
        SELECT max(age) INTO max_age FROM student;
        DBMS_OUTPUT.PUT_LINE('学生最大年龄为:'||max_age);
    END t2;
```

实验 2　行级触发器

（1）

```
CREATE OR REPLACE TRIGGER t3
AFTER UPDATE OF salary ON emp
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('该员工的姓名为:'||:old.ename);
    DBMS_OUTPUT.PUT_LINE('修改前的工资为:'||:old.salary);
    DBMS_OUTPUT.PUT_LINE('修改后的工资为:'||:new.salary);
END t3;
```

（2）

```
CREATE OR REPLACE TRIGGER t4
BEFORE DELETE ON dept
FOR EACH ROW
BEGIN
    DELETE FROM emp WHERE deptno = :old.deptno;
END t4;
```

第13章实验

实验1 用户管理

(1)

```
CREATE USER test2
IDENTIFIED BY test2
DEFAULT TABLESPACE system
QUOTA 10M ON system;
```

不能立即登录,原因是创建用户后,必须给该用户授权,用户才能连接到数据库,并对数据库中的对象进行操作。只有拥有CREATE SESSION权限的用户才能连接到数据库。

(2)

```
CREATE USER test3
IDENTIFIED BY test3
DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp
QUOTA 20M ON users
PASSWORD EXPIRE
ACCOUNT LOCK;
```

(3)

```
ALTER USER test3
IDENTIFIED BY tiger
DEFAULT TABLESPACE system
TEMPORARY TABLESPACE temp
ACCOUNT UNLOCK
```

(4)

```
CREATE USER test4
IDENTIFIED BY test4
ACCOUNT LOCK;
```

(5)

```
DROP USER test4;
```

实验 2 权限管理

（1）

```
GRANT CREATE SESSION,CREATE TABLE,CREATE VIEW
TO test2
WITH ADMIN OPTION;
```

（2）

```
GRANT CREATE TABLE
TO test3
```

（3）

```
REVOKE CREATE VIEW
FROM test2;
```

（4）

```
GRANT SELECT,INSERT
ON emp
TO test2;
```

（5）

```
REVOKE INSERT
ON emp
FROM test2;
```

实验 3 角色管理

（1）

```
CREATE ROLE emp_role;
```

（2）

```
GRANT SELECT,UPDATE
ON emp
TO emp_role;
```

（3）

```
GRANT emp_role TO scott;
```

（4）

```
REVOKE emp_role FROM scott;
```

(5)

```
DROP ROLE emp_role;
```

第14章实验

实验1 数据库的备份

(1) 采用交互方式,使用EXP逻辑备份

参考例题14.4。

(2) 使用命令行方式备份

参考例题14.5。

(3) 使用参数文件方式备份

参考例题14.6。

实验2 数据库的恢复

(1) 采用交互方式,使用IMP逻辑恢复

参考例题14.8。

(2) 使用命令行方式恢复

参考例题14.9。

(3) 使用参数文件方式恢复

参考例题14.10。

附录 D

课后习题参考答案

第 1 章

一、1. B 2. D 3. A 4. D 5. B 6. C 7. A 8. C 9. C 10. B

二、1. 数据库(DataBase,DB):数据库是长期储存在计算机内的、有组织的、可共享的数据集合。数据库中的数据按一定的数据模型组织、描述和储存,具有较小的冗余度、较高的数据独立性和易扩展性,并可为各种用户共享。

数据库系统(DataBase System,DBS):数据库系统是指在计算机系统中引入数据库后的系统构成,一般由数据库、数据库管理系统(及其开发工具)、应用系统、数据库管理员构成。数据库系统和数据库是两个概念,数据库系统是一个人-机系统,数据库是数据库系统的一个组成部分。

数据库管理系统(DataBase Management System,DBMS):数据库管理系统是位于用户与操作系统之间的一层数据管理软件,用于科学地组织和存储数据、高效地获取和维护数据。数据库管理系统的主要功能包括数据定义功能、数据操纵功能、数据控制功能、数据库的运行管理功能、数据库的建立和维护功能。数据库管理系统是一个大型的复杂的软件系统,是计算机中的基础软件。

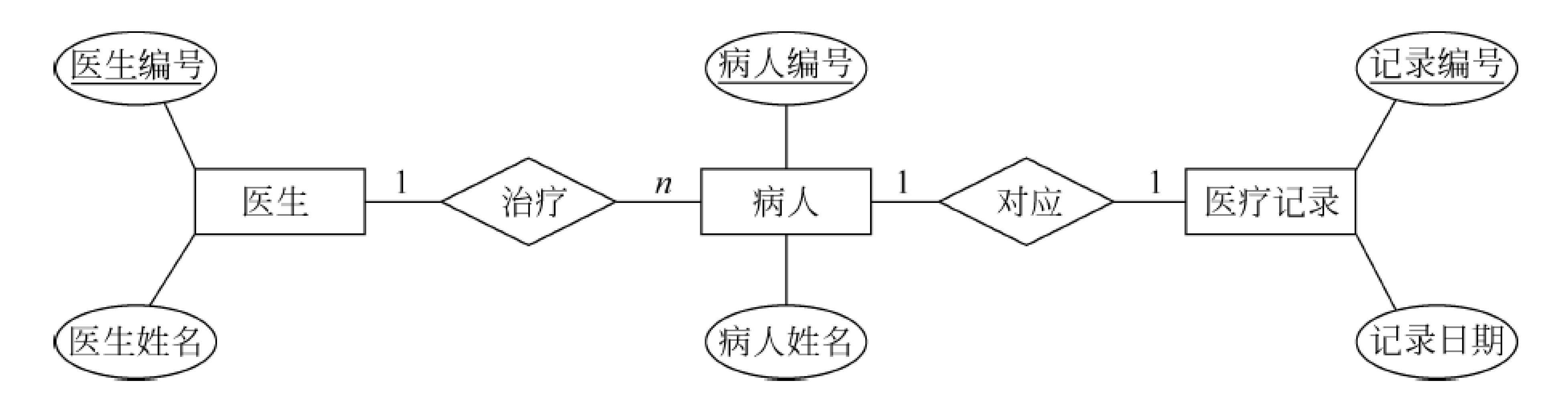
2. 目前通用的关系型数据库有 Oracle、SQL Server、MySQL、DB2 等。

3. 数据与程序的逻辑独立性:当模式改变时(例如增加新的关系、新的属性、改变属性的数据类型等),由数据库管理员对各个外模式/模式的映像做相应改变,可以使外模式保持不变。应用程序是依据数据的外模式编写的,从而应用程序不必修改,保证了数据与程序的逻辑独立性,简称数据的逻辑独立性。

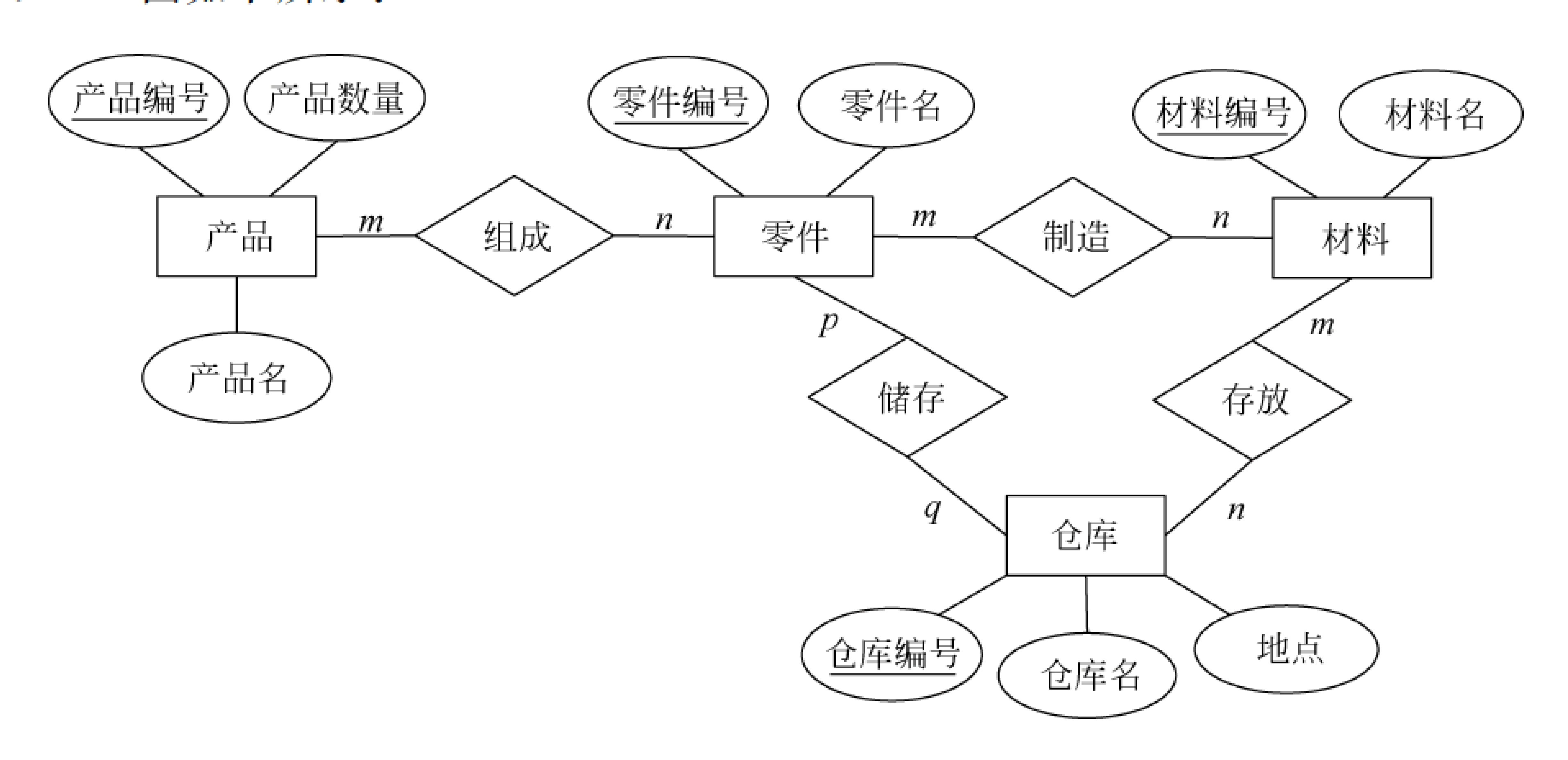
数据与程序的物理独立性:当数据库的存储结构改变了,由数据库管理员对模式/内模式映像做相应改变,可以使模式保持不变,从而应用程序也不必改变,保证了数据与程序的物理独立性,简称数据的物理独立性。

数据库管理系统在三级模式之间提供的两级映像保证了数据库系统中的数据能够具有较高的逻辑独立性和物理独立性。

三、1. E-R 图如下所示：



2. E-R 图如下所示：



第 2 章

一、1. B 2. C 3. D 4. C 5. B 6. D 7. C 8. A 9. D 10. C

二、1. 关系 R 和关系 S 交集的共同部分假定设为 T。关系 R 中属于 R 而不属于 S 的部分是 $(R-S)$，而关系 S 中属于 S 但不属于 R 的部分是 $(S-R)$。此时，在关系 R 中得到 $T=(R-(R-S))$，在关系 S 中得到 $T=(S-(S-R))$。所以，$R\cap S=(R-(R-S))=(S-(S-R))$。由此可知，关系的交运算可以用差运算来表示。

2. 针对传统的集合运算 $R\cup S$、$R\cap S$、$R-S$、$R\bowtie S$ 中可能出现的几种极端情况有：R 和 S 完全没有交点；S 完全包含在 R 中。根据极端情况我们可以得出如下答案。

$R\cap S$：最小元组数量 0，最大元组数量 m

$R\cup S$：最小元组数量 m，最大元组数量 $(m+n)$

$R-S$：最小元组数量 $(m-n)$，最大元组数量 m

$R\bowtie S$：最小元组数量0，最大元组数量$(m*n)$

3. 关系模型中有三类完整性约束：实体完整性、参照完整性和用户定义的完整性。

实体完整性规则：若属性A是基本关系R的主属性，则属性A不能取空值。实体完整性规则规定：基本关系的所有主属性都不能取空值。

参照完整性规则：若属性(或属性组)F是基本关系R的外码，它与基本关系S的主码K_S相对应(基本关系R和S不一定是不同的关系)，则对于R中每个元组在F上的值必须为：或者取空值(F的每个属性值均为空值)；或者等于S中某个元组的主码值。参照完整性规则规定：关系R和关系S不一定是不同的关系，F与K_S必须定义在同一域中。

用户定义完整性规则：不同的关系数据库系统根据其应用环境的不同，往往需要一些特殊的约束条件，这就是用户定义的完整性约束条件。用户定义完整性规则规定：针对某一具体关系数据库的约束条件，反映某一具体应用所涉及的数据必须满足的语义要求。

4. 连接运算符是"="的连接运算称为等值连接。它是从关系R与S的广义笛卡儿积中选取A、B属性值相等的那些元组。自然连接是一种特殊的等值连接，它要求两个关系中进行比较的分量必须是相同的属性组，并且在结果中把重复的属性列去掉。

三、关系代数运算表达式如下：

(1) $\pi_{address}(\sigma_{sname='旺达超市'}(shop))$

(2) $\pi_{cid}(\sigma_{price>=100 \wedge price<=150}(commodity))$

(3) $\pi_{sname}(\sigma_{cname='旺仔牛奶'}(shop \bowtie sale \bowtie commodity))$

(4) $\pi_{sid}(shop)-\pi_{sid}(\sigma_{price>5000}(commodity \bowtie sale))$

(5) $\pi_{1}(\sigma_{2='C1' \wedge 5='C2'}(sale \times sale))$

第3章

一、1. D 2. C 3. B 4. A 5. C 6. C 7. B 8. D 9. A 10. D

二、1. 建立索引的目的是为了快速查询，减少查询操作的时间。建立索引的个数不是越多越好，如果数据增、删、改频繁，系统就会花费许多时间来维护索引，过多的索引会导致索引碎片，降低系统效率，因此不必要的索引应该及时删除。

2. WHERE子句的作用是在分组之前过滤数据。WHERE条件中不能包含聚集函数。使用WHERE条件选择满足条件的行。

HAVING子句的作用是在分组之后过滤数据。HAVING条件中经常包含聚集函数。使用HAVING条件选择满足条件的组。使用HAVING子句时必须首先使用GROUP BY进行分组。

3. 基本表是本身独立存在的表，在SQL中一个关系对应一个表。视图是从一个或多个基本表中导出的表。

视图本身并不独立存储在数据库中，是一个虚表。数据库中只存放视图的定义而不存

放其所对应的数据，这些数据仍然存放在导出视图的基本表中。用户可以像使用基本表那样使用视图，可以在视图上再定义视图。

三、1.（1）用 SQL 语句实现以下基本表的创建

① CREATE TABLE book
```
(bno CHAR(10) PRIMARY KEY,
bname VARCHAR(20) NOT NULL,
author VARCHAR(20),
price NUMBER CHECK(price>0));
```

② CREATE TABLE reader
```
(rno CHAR(10) PRIMARY KEY,
rname VARCHAR(20) NOT NULL,
address VARCHAR(50));
```

③ CREATE TABLE br
```
(bno CHAR(10) REFERENCES book(bno),
rno CHAR(10) REFERENCES reader(rno),
datetime DATE,
PRIMARY KEY(bno,rno));
```

（2）根据各表结构，用 SQL 语句完成下列操作

① ALTER TABLE book MODIFY bname VARCHAR(20);

② ALTER TABLE reader ADD age NUMBER;

③ UPDATE book SET price = 24 WHERE bno = 'b1';

④ DELETE FROM br WHERE rno IN (SELECT rno FROM reader WHERE rname = '张三');

⑤ SELECT COUNT(*) FROM book WHERE price > 100;

⑥ SELECT bno, price FROM book WHERE bname LIKE '%数据库%';

⑦ SELECT rname, datetime FROM book, br, reader WHERE reader.rno = br.rno AND book.bno = br.bno AND bname = '时间简史';

⑧ SELECT rname FROM reader WHERE rno IN (SELECT rno FROM br GROUP BY rno HAVING COUNT(*)> 5);

⑨ SELECT rno, rname FROM reader WHERE NOT EXISTS (SELECT * FROM br WHERE reader.rno = br.rno);

⑩ SELECT bname, price FROM book WHERE price = (SELECT MAX(price) FROM book);

2.（1）用 SQL 语句实现以下基本表的创建

① CREATE TABLE s
```
(sno CHAR(10) PRIMARY KEY,
sn VARCHAR(20) NOT NULL,
city VARCHAR(20));
```

② CREATE TABLE p
```
(pno CHAR(10) PRIMARY KEY,
pn VARCHAR(20) NOT NULL,
```

```
   color VARCHAR(20),
   weight NUMBER CHECK(weight > 0));
③ CREATE TABLE j
   (jno CHAR(10) PRIMARY KEY,
   jn VARCHAR(20) NOT NULL,
   city VARCHAR(20));
④ CREATE TABLE spj
   (sno CHAR(10) REFERENCES s(sno),
   pno CHAR(10) REFERENCES p(pno),
   jno CHAR(10) REFERENCES j(jno),
   qty INT,
   PRIMARY KEY(sno,pno,jno));
```

(2) 根据各表结构,用 SQL 语句完成下列操作

① SELECT pn,color,weight FROM p;

② SELECT * FROM j WHERE city = '大连';

③ SELECT pn FROM p WHERE weight = (SELECT MIN(weight) FROM p);

④ SELECT sn FROM s,spj WHERE spj.sno = s.sno AND jno = 'j1';

⑤ SELECT distinct color FROM p,spj WHERE p.pno = spj.pno AND sno = 's1';

⑥ SELECT pn,qty FROM p,spj WHERE p.pno = spj.pno AND jno = 'j2';

⑦ SELECT sn FROM s, spj, p WHERE s. sno = spj. sno AND p. pno = spj. pno AND p. pno = 'p2' AND color = '绿色' AND qty > 500;

⑧ SELECT city,count(*)FROM s GROUP BY city ORDER BY count(*) DESC;

⑨ SELECT jno FROM j WHERE NOT EXISTS(SELECT * FROM s,spj WHERE s.sno = spj.sno AND j.jno = spj.jno AND city = '大连';

⑩ SELECT sno FROM spj WHERE jno = 'j1' INTERSECT SELECT sno FROM spj WHERE jno = 'j2';

第 4 章

一、1. B　2. B　3. C　4. A　5. C　6. D　7. D　8. C　9. C　10. C

二、1. 对于关系模式 $R\in$2NF,若它的每一个非主属性不传递函数依赖于 R 的候选码,则关系模式 $R\in$3NF。因此,2NF 的关系模式消除了非主属性对候选码的传递函数依赖后就变成了 3NF 的,而 3NF 的关系模式必定是 2NF 和 1NF 的。

2. E-R 图为实体—联系图,提供了表示实体、属性和联系的方法,用来描述现实世界的概念模型。

构成 E-R 图的基本要素是实体、属性和联系,其表示方法为:

实体:用矩形表示,矩形框内写明实体名;

属性：用椭圆形表示，并用无向边将其与相应的实体连接起来；

联系：用菱形表示，菱形框内写明联系名，并用无向边分别与有关实体连接起来，同时在无向边旁标上联系的类型（1∶1、1∶n 或 m∶n）。

3. 一个实体型转换为一个关系模式。实体的属性就是关系的属性，实体的码就是关系的码。

对于实体间的联系则有以下不同的情况：

(1) 一个 1∶1 联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，每个实体的码均是该关系的候选码。如果与某一端实体对应的关系模式合并，则需要在该关系模式的属性中加入另一个关系模式的码和联系本身的属性。

(2) 一个 1∶n 联系可以转换为一个独立的关系模式，也可以与 n 端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而该关系的码为 n 端实体的码。如果与 n 端对应的关系模式合并，则只需要将联系本身的属性和 1 端实体的码加入到 n 端对应的关系模式中即可。

(3) 一个 m∶n 联系转换为一个独立的关系模式。与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，该关系的码是各实体码的组合。

(4) 三个或三个以上实体间的一个多元联系可以转换为一个关系模式。与该多元联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为各实体码的组合。

(5) 同一实体集的实体间的联系，即自联系，也可按 1∶1、1∶n 和 m∶n 三种情况分别处理。

(6) 具有相同码的关系模式可合并。

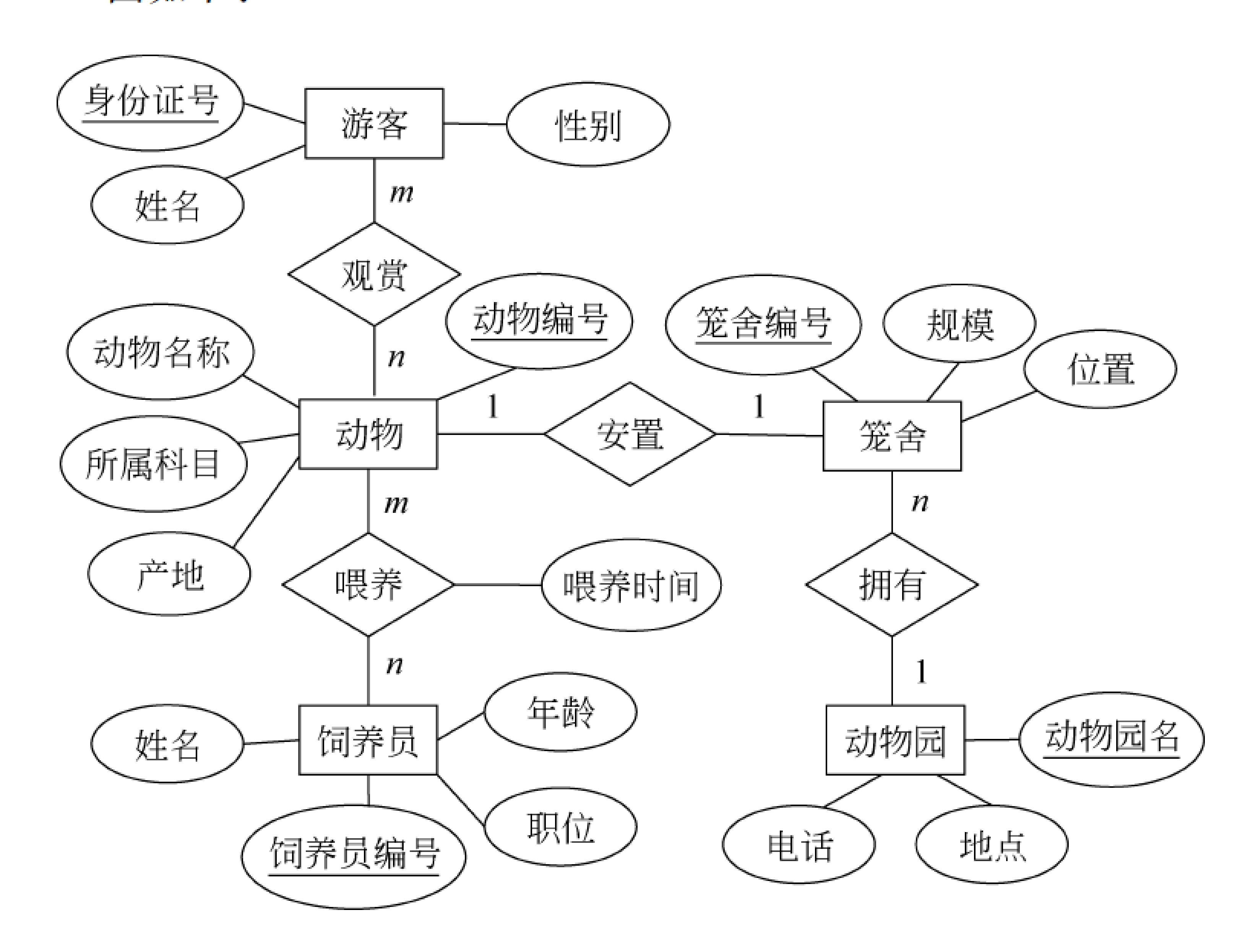
三、1. (1) 根据上述语义，有三个函数依赖：(商店编号，商品编号)→部门编号；(商店编号，部门编号)→部门经理；(商店编号，商品编号)→库存数量。

(2) 关系模式 R 的候选码是(商店编号，商品编号)。

(3) 因为关系模式 R 中存在非主属性“部门经理”传递函数依赖于候选码(商店编号，商品编号)，所以 R 满足 2NF，R 不满足 3NF。

(4) 将关系模式 R 分解得：$R1$(商店编号，商品编号，库存数量，部门编号)；$R2$(商店编号，部门编号，部门经理)。此时，$R1$ 和 $R2$ 都满足 3NF。

2.（1）E-R 图如下：



（2）关系模式如下：

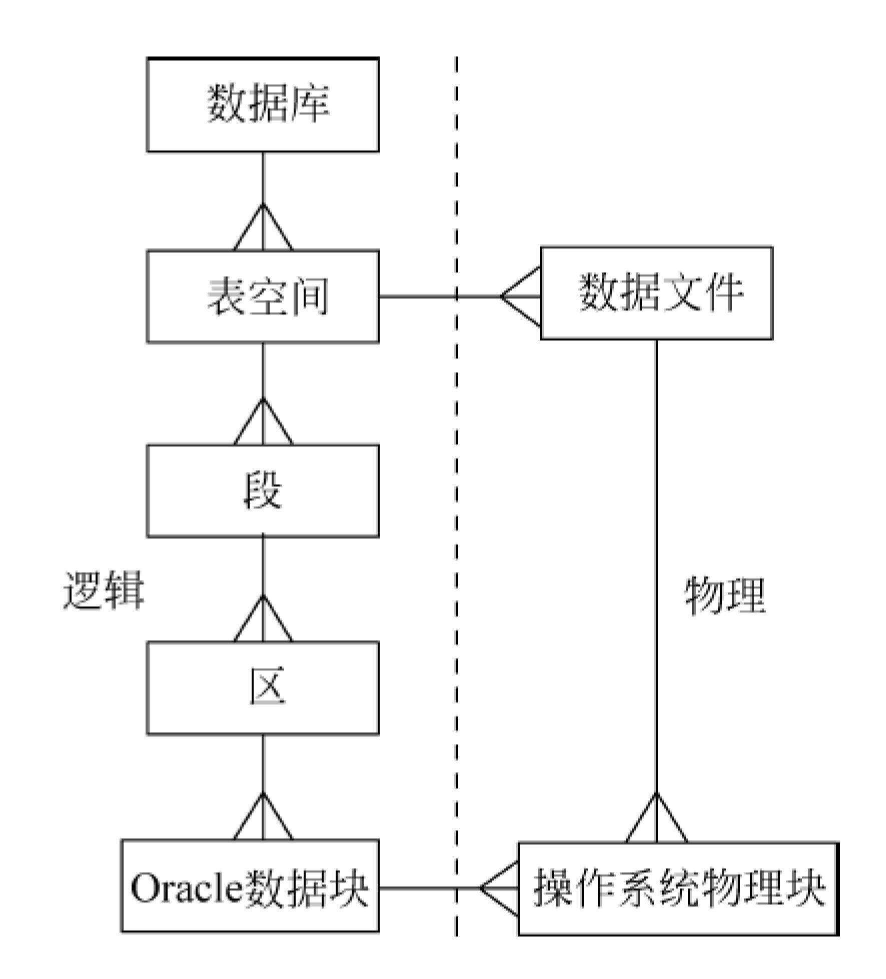
动物园(动物园名,地点,电话)
笼舍(笼舍编号,规模,位置,动物园名,动物编号)
动物(动物编号,动物名称,所属科目,产地)
饲养员(饲养员编号,姓名,年龄,职位)
游客(身份证号,姓名,性别)
观赏(身份证号,动物编号)
喂养(动物编号,饲养员编号,喂养时间)

第 5 章

一、1. D　2. C　3. A　4. D　5. B

二、1. Oracle 数据库逻辑结构和物理结构的关系见下图。

一个数据库在逻辑上有多个表空间组成,一个表空间又可以由多个段组成,一个段由若干个区组成,区又是由一组连续的 Oracle 数据块组成。一个表空间可以跨越多个物理结构中的数据文件,一个数据文件只能属于一个表空间所有。一个数据文件由多个操作系统物理块所组成。Oracle 块是输入或输出的最小单位,一般由一个或多个操作系统块组成。



2. 系统全局区的数据被多个用户共享。当数据库启动时,系统全局区内存被自动分配。它主要包含以下几个内存结构:共享池(Shared Pool)、数据库高速缓冲区(Database Buffer Cache)、重做日志缓冲区(Redo Log Buffer)和其他的一些结构等。

共享池由库高速缓存(Library Cache)和数据字典高速缓存(Data Dictionary Cache)两部分所组成。系统将 SQL(也可能是 PL/SQL)语句的正文和编译后的代码以及执行计划都放在共享池的库高速缓存中。

数据库高速缓冲区主要目的是为了缓存操作的数据,从而减少系统读取磁盘的次数。

重做日志缓冲区的主要目的就是数据的恢复。Oracle 系统在使用任何 DML 或 DDL 操作时,会先把该操作的信息写入重做日志缓冲区中,之后才对数据高速缓冲区的内容进行修改。

第 6 章

一、1. D 2. B 3. B 4. A 5. C

二、1.

```
DECLARE
    start_range NUMBER: = &start;
    end_range NUMBER: = &end;
BEGIN
    FOR i IN REVERSE start_range..end_range LOOP
        DBMS_OUTPUT.PUT_LINE('i = '||i);
    END LOOP;
END;
```

2.

使用 IF 语句实现：

```
DECLARE
    v_sno student.sno%TYPE: = &sno;
    v_grade sc.grade%TYPE;
BEGIN
    SELECT AVG(grade) INTO v_grade FROM sc WHERE sno = v_sno;
    IF v_grade >= 90 THEN
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为优秀!');
    ELSIF v_grade >= 80 THEN
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为良好!');
    ELSIF v_grade >= 70 THEN
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为中等!');
    ELSIF v_grade >= 60 THEN
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为及格!');
    ELSE
        DBMS_OUTPUT.PUT_LINE('此同学平均成绩为不及格!');
    END IF;
END;
```

使用 CASE 语句实现：

```
DECLARE
    v_sno student.sno%TYPE: = &sno;
    v_grade sc.grade%TYPE;
BEGIN
    SELECT AVG(grade) INTO v_grade FROM sc WHERE sno = v_sno;
    CASE
        WHEN v_grade >= 90 THEN
            DBMS_OUTPUT.PUT_LINE('此同学平均成绩为优秀!');
        WHEN v_grade >= 80 THEN
            DBMS_OUTPUT.PUT_LINE('此同学平均成绩为良好!');
        WHEN v_grade >= 70 THEN
            DBMS_OUTPUT.PUT_LINE('此同学平均成绩为中等!');
        WHEN v_grade >= 60 THEN
            DBMS_OUTPUT.PUT_LINE('此同学平均成绩为及格!');
        ELSE
            DBMS_OUTPUT.PUT_LINE('此同学平均成绩为不及格!');
    END CASE;
END;
```

第 7 章

一、1. A　2. D　3. A　4. C　5. D

二、自定义异常处理分为三步：

(1) 定义异常。

在 DECLARE 部分定义异常,异常的类型为 EXCEPTION。

(2) 触发异常。

在 BGEIN 部分,当一个设定条件满足时,可以显式通过 RAISE 语句来触发自定义异常。

(3) 异常处理。

在 EXCEPTION 部分处理,和系统预定义异常的处理方式一致。如果没有 EXCEPTION 部分,则由系统处理异常。

三、1.

```
DECLARE
    num NUMBER ;
    e_few EXCEPTION;
    e_many EXCEPTION;
BEGIN
    SELECT  COUNT(sno) INTO num FROM sc WHERE cno = 'c1';
    DBMS_OUTPUT.PUT_LINE('选修人数: '||num);
    IF num < 10 THEN
        RAISE e_few;
    ELSIF num > 50 THEN
        RAISE e_many;
    END IF;
EXCEPTION
    WHEN e_few THEN
        DBMS_OUTPUT.PUT_LINE('选修人数太少,无法开课');
    WHEN e_many THEN
        DBMS_OUTPUT.PUT_LINE('选修人数太多,需增加授课教师');
END;
```

2.

```
DECLARE
    v_sal   emp.salary%TYPE;
    v_eno   emp.empno%TYPE: = &v_empno;
    e EXCEPTION;
BEGIN
    SELECT salary INTO v_sal FROM emp WHERE  empno = v_eno;
```

```
        CASE
          WHEN v_sal < 500 THEN
              UPDATE emp SET salary = salary * 1.5 WHERE empno = v_eno;
          WHEN (v_sal BETWEEN 500  AND  1500 ) THEN
              UPDATE emp SET salary = salary * 1.3 WHERE empno = v_eno;
          WHEN (v_sal BETWEEN 1500 AND  3000) THEN
              UPDATE emp SET salary = salary * 1.1 WHERE empno = v_eno;
          WHEN (v_sal > 3000) THEN
              RAISE e;
          END CASE;
EXCEPTION
              WHEN e THEN
                  DBMS_OUTPUT.PUT_LINE('工资太高,不需涨薪');
END;
```

第 8 章

一、1. C　2. C　3. C　4. D　5. B

二、1. 显式游标的处理分为定义游标、打开游标、取值到变量和关闭游标四个步骤。

(1) 定义游标：定义游标时需要定义游标的名字，并将该游标和一个 SELECT 语句相关联。

(2) 打开游标：当打开游标时，Oracle 会执行游标所对应的 SELECT 语句，并且将 SELECT 语句的结果暂时存放到内存的数据缓冲区中。

(3) 取值到变量：在打开游标之后，SELECT 语句的结果被临时存放到游标结果集中。为了处理结果集中的数据，需要使用 FETCH 语句提取游标数据。

(4) 关闭游标：当提取并处理了结果集的所有数据之后，就可以关闭游标并释放其结果集。

2.

隐式游标	显式游标
由 PL/SQL 维护，当执行查询时自动打开和关闭	需要在程序中显式定义、打开、关闭，游标有一个名字
游标属性前缀是 SQL	游标属性的前缀是用户自定义的游标名
属性%ISOPEN 总是为 FALSE	%ISOPEN 根据游标的状态来确定值
SELECT 语句带有 INTO 子串，只有一行数据被处理	可以处理多行数据，在程序中设置循环，取出每一行数据

三、1.

```
DECLARE
```

```
    CURSOR c1 IS SELECT sname,sex FROM student WHERE age BETWEEN 18 AND 20;
BEGIN
    FOR a IN c1 LOOP
       DBMS_OUTPUT.PUT_LINE(a.sname||','||a.sex);
    END LOOP;
END;
```

2.

```
DECLARE
    CURSOR  emp_cursor(v_dname CHAR)  IS  SELECT empno,ename  FROM emp,dept WHERE
emp.deptno = dept.deptno AND dname = v_dname;
BEGIN
    FOR  m  IN emp_cursor('财务部') LOOP
        DBMS_OUTPUT.PUT_LINE(m.empno||','||m.ename);
    END LOOP;
END;
```

第 9 章

一、1. A　2. B　3. A

二、1.

(1)

```
CREATE OR REPLACE PROCEDURE p1
(v_empno IN emp.empno%TYPE,
v_ename OUT emp.ename%TYPE,
v_salary  OUT emp.salary%TYPE)
IS
BEGIN
    SELECT ename,salary INTO v_ename,v_salary FROM emp WHERE empno = v_empno;
END p1;
```

(2)

```
DECLARE
    s_ename emp.ename%TYPE;
    s_salary emp.salary%TYPE;
BEGIN
    p1('2001',s_ename,s_salary);
    DBMS_OUTPUT.PUT_LINE(s_ename||','||s_salary);
END;
```

2.

(1)

```
CREATE OR REPLACE PROCEDURE p2
(v_empno IN emp.empno%TYPE,
v_ename IN emp.ename%TYPE)
IS
    e EXCEPTION;
BEGIN
    UPDATE emp SET ename = v_ename WHERE empno = v_empno;
    IF SQL%NOTFOUND THEN
        RAISE e;
    END IF;
EXCEPTION
    WHEN e THEN
        DBMS_OUTPUT.PUT_LINE('不存在该员工!');
END p2;
```

(2)

```
BEGIN
    p2('1001','天天');
END;
```

3.

(1)

```
CREATE OR REPLACE PROCEDURE p3
(v_sno IN sc.sno%TYPE)
IS
    CURSOR c_sc IS SELECT cno,grade FROM sc WHERE sno = v_sno;
BEGIN
    FOR v_sc IN c_sc LOOP
        DBMS_OUTPUT.PUT_LINE(v_sc.cno||','||v_sc.grade);
    END LOOP;
END p3;
```

(2)

```
BEGIN
    p3('11432002');
END;
```

第 10 章

一、1. B　2. C　3. C　4. B

二、1.

(1)

```
CREATE OR REPLACE FUNCTION f1(v_empno IN emp.empno%TYPE)
RETURN emp%ROWTYPE
IS
    v_emp emp%ROWTYPE;
BEGIN
    SELECT * INTO v_emp FROM emp WHERE empno = v_empno;
    RETURN v_emp;
END f1;
```

(2)

```
DECLARE
    s_emp emp%ROWTYPE;
BEGIN
    s_emp:= f1('3002');
    DBMS_OUTPUT.PUT_LINE(s_emp.ename||','||s_emp.salary);
END;
```

2.

(1)

```
CREATE OR REPLACE FUNCTION f2
(v_sno IN sc.sno%TYPE)
RETURN NUMBER
IS
    v_num  NUMBER;
BEGIN
    SELECT AVG(grade) INTO v_num  FROM sc WHERE sno = v_sno;
    RETURN v_num;
END f2;
```

(2)

```
DECLARE
    v_num NUMBER;
BEGIN
    v_num:= f2('11432001');
    DBMS_OUTPUT.PUT_LINE(v_num);
END;
```

第 11 章

一、1. B 2. A 3. C

二、1.

（1）

包说明的创建：

```
CREATE OR REPLACE PACKAGE pkg1
IS
    FUNCTION return_avgsal(v_deptno emp.deptno%TYPE)
    RETURN emp.salary%TYPE;
    PROCEDURE show_emp(v_deptno emp.deptno%TYPE);
END pkg1;
```

包主体的创建：

```
CREATE OR REPLACE PACKAGE BODY pkg1
IS
    FUNCTION return_avgsal(v_deptno emp.deptno%TYPE)
    RETURN emp.salary%TYPE
    IS
        v_avgsal emp.salary%TYPE;
    BEGIN
        SELECT AVG(salary) INTO v_avgsal FROM emp WHERE deptno = v_deptno;
    RETURN v_avgsal;
    END return_avgsal;
    PROCEDURE show_emp(v_deptno emp.deptno%TYPE)
    IS
        CURSOR c_emp IS SELECT * FROM emp WHERE deptno = v_deptno AND
salary<return_avgsal(v_deptno);
    BEGIN
        FOR v_emp IN c_emp LOOP
        DBMS_OUTPUT.PUT_LINE('低于平均工资的员工信息为：'|| v_emp.empno||','||v_emp.
ename);
        END LOOP;
    END show_emp;
END pkg1;
```

（2）

```
DECLARE
s_avgsal emp.salary%TYPE;
BEGIN
s_avgsal:= pkg1.return_avgsal('30');
DBMS_OUTPUT.PUT_LINE('该部门平均工资为'||s_avgsal);
pkg1.show_emp('30');
END;
```

2.

(1)

包说明的创建：

```
CREATE OR REPLACE PACKAGE overload_package
AS
    PROCEDURE get_course(v_cno char);
    PROCEDURE get_course(v_credit course.credit%TYPE);
END overload_package;
```

包主体的创建：

```
CREATE OR REPLACE PACKAGE BODY overload_package
AS
    PROCEDURE get_course(v_cno char)
    AS
        v_cou course%ROWTYPE;
    BEGIN
        SELECT * INTO v_cou FROM course WHERE cno = v_cno;
        DBMS_OUTPUT.PUT_LINE('课程名称为:'||v_cou.cname||',授课教师为: '||v_cou.tname||',
学分为:'||v_cou.credit);
    END get_course;
    PROCEDURE get_course(v_credit course.credit%TYPE)
    AS
        v_cou course%ROWTYPE;
        CURSOR c_cou IS SELECT * FROM course WHERE credit = v_credit;
    BEGIN
        FOR v_cou IN c_cou LOOP
            DBMS_OUTPUT.PUT_LINE('课程名称为:'||v_cou.cname||',授课教师为: '||v_cou.tname|
|',学分为:'||v_cou.credit);
        END LOOP;
    END get_course;
END overload_package;
```

(2)

```
BEGIN
    overload_package.get_course('c4');
END;
```

(3)

```
BEGIN
    overload_package.get_course(3);
END;
```

第 12 章

一、1. B　2. B　3. C

二、1.

```
CREATE OR REPLACE TRIGGER t1
AFTER INSERT OR DELETE ON student
DECLARE
    v_count NUMBER;
    v_age student.age%TYPE;
BEGIN
    IF INSERTING THEN
        SELECT AVG(age) INTO v_age FROM student;
        DBMS_OUTPUT.PUT_LINE('学生的平均年龄为:'||v_age);
    END IF;
    IF DELETING THEN
        SELECT COUNT(*) INTO v_count FROM student WHERE sex = '女';
        DBMS_OUTPUT.PUT_LINE('女学生的人数为:'||v_count);
    END IF;
END t1;
```

2.

```
CREATE OR REPLACE TRIGGER t2
BEFORE DELETE ON student
FOR EACH ROW
BEGIN
    UPDATE sc SET sno = :new.sno WHERE sno = :old.sno;
END t2;
```

3.

```
CREATE OR REPLACE TRIGGER t3
BEFORE UPDATE OF salary ON emp
FOR EACH ROW
WHEN((old.deptno = 30 AND new.salary > 8000) OR (old.deptno!= 10 AND new.salary > 10000))
BEGIN
    RAISE_APPLICATION_ERROR(-20001,'工资更新超出范围!');
END t3;
```

第 13 章

一、1. B　2. C　3. A　4. C　5. B

二、1.

```
CREATE USER test_user
IDENTIFIED BY tiger
DEFAULT TABLESPACE users
ACCOUNT LOCK;
```

2.

```
ALTER USER test_user
ACCOUNT UNLOCK;
```

3.

```
GRANT SELECT,DELETE
ON course
TO test_user
WITH GRANT OPTION;
```

4.

```
GRANT UPDATE(cname)
ON course
TO test_user;
```

5.

```
REVOKE DELETE
ON course
FROM test_user;
```

6.

```
CREATE ROLE test_role;
```

7.

```
GRANT SELECT,UPDATE
ON dept
TO test_role;
```

8.

```
GRANT test_role TO test_user;
```

9.

```
REVOKE test_role FROM PUBLIC;
```

10.

```
DROP ROLE test_role;
```

11.

```
DROP USER test_user;
```

第 14 章

一、1. A　2. B　3. C　4. C　5. D

二、(1)

```
c:\EXP USERID = system/密码
FILE = e:\orcl\course.dmp
TABLES = (course);
```

(2)

```
c:\IMP USERID = system/密码
TABLES = (course)
ROWS = Y
FILE = e:\orcl\course.dmp;
```

参 考 文 献

[1] 王珊，萨师煊. 数据库系统概论[M]. 第四版. 北京：高等教育出版社，2006.
[2] 张迎新. 数据库原理、方法与应用[M]. 北京：高等教育出版社，2004.
[3] 郑铃利. 数据库原理与应用案例教程[M]. 第 2 版. 北京：清华大学出版社，2013.
[4] 王珊. 数据库系统概论学习指导与习题解析[M]. 北京：高等教育出版社，2008.
[5] 王雅轩. 实用数据库原理与应用[M]. 沈阳：辽宁人民出版社，2011.
[6] 钱慎一. Oracle 11g 数据库基础与应用教程[M]. 北京：清华大学出版社，2011.
[7] 杨晨，闫薇. Oracle 数据库应用教学做一体化教程[M]. 北京：清华大学出版社，2013.
[8] Patrice Daux，Jeff Gallus，Jan Speelpenning. Data Modeling and Relational Database Design[M]. Oracle Corporation，2002.
[9] 张红强. Oracle 数据库实例教程[M]. 天津：天津大学出版社，2009.
[10] 陆云帆. Oracle 数据库设计与实现[M]. 北京：机械工业出版社，2011.